우리식조작체계문고

# JAVA 프로그람련습문제집

교육성교육정보쎈터
주체97(2008)

# 차 례

# 머 리 말

위대한 령도자 **김정일**동지께서는 다음과 같이 지적하시였다.

**《프로그람을 개발하는데서 기본은 우리 식의 프로그람을 개발하는것입니다.》**

(《**김정일**선집》 제15권, 196페지)

위대한 령도자 **김정일**동지의 현명한 령도에 의하여 오늘 우리 나라에서는 정보산업이 비약적으로 발전하고있다. 더우기 콤퓨터망이 전국적범위에서 형성됨으로써 정보기술, 프로그람기술이 과학연구뿐아니라 생산과 경영활동을 비롯한 사회생활의 모든 분야에 광범히 도입되여 응용되고있으며 경제적효과성을 높이고있다.

이런데로부터 Java언어는 많은 프로그람개발자들과 과학자, 기술자들이 관심을 가지고 활발히 활용하고있는 망기반의 프로그람작성언어이다.

Java언어의 학습에서 실천은 매우 중요하다. 눈에는 익고 손에는 설다고 책으로 보면 리해가 되고 얼마든지 활용할수 있다고 생각되지만 현실에 부딪쳐 해보자고 하면 잘 되지 않는것이 바로 실천이다. 더우기 Java언어는 현실세계에서 다루는 모든 사물과 개념적형태들을 개념세계의 개념적모형화를 거쳐 콤퓨터세계로 표현하여야 하는것만큼 실천을 통하여 지식을 공고히 하여야 원만히 활용할수 있다.

이 책은 Java언어와 객체지향프로그람설계를 학습하는 사람들이 실천을 통하여 빠른 기간에 높은 수준의 설계능력을 소유하도록 하기 위하여 련습문제들을 선택하여 묶은것이다. 콤퓨터로 프로그람을 실행하여 검증하였을 때에 비로소 문제를 정확하게 풀었다고 말할수 있다. 그러므로 Java언어를 배우는 독자들은 자체로 프로그람을 작성하고 콤퓨터상에서 실행시켜보면서 이 책의 풀이와 비교도 하고 수정도 하면서 공부하면 빨리 정확한 리해에 도달하게 될것이다.

이 책은 장마다 련습문제뒤에 보충문제를 주어 장의 내용을 깊이 있게 파악하도록 편성되여있다. 또한 련습문제는 쉬운 문제로부터 시작하여 점차 어려운 문제로 올라가는식으로 구성하였는데 쉽게 파악하는 사람들은 건너뛰면서 학습할수도 있다. 이 책에 있는 련습문제를 다 풀어본 다음에는 요구에 따라 프로그람들을 변경하면서 프로그람설계개념과 프로그람작성기술, 응용측면에 대하여 더 깊이 학습하여야 한다.

# 제1장. Java프로그람작성초보

## 1.1. 련습

**1.1.1. 수속지향문제풀이와 객체지향문제풀이의 다른점을 간단히 서술하고 수속지향과 객체지향의 프로그람작성언어를 각각 2가지씩 드시오.**

수속지향문제풀이는 콤퓨터가 리해할수 있는 리산론리를 리용하여 풀려고 하는 문제자체와 구체적인 문제해결과정을 서술하고 표현한다. 여기서 기본은 알고리듬과 자료구조이다.

객체지향문제풀이는 사람들의 일상적인 사고에 부합되는 방식을 리용하여 풀려고 하는 문제, 그의 구조, 특징, 여러가지 동적행위를 모형화하여 문제풀이의 답을 얻는다.

여기서 기본은 클라스, 객체, 설계형식이다.

수속지향프로그람작성언어는 BASIC, FORTRAN, Pascal, C 등이다.

객체지향프로그람작성언어는 Smalltalk-80, Object Pascal, C++, Java 등이다.

**1.1.2. 객체, 클라스, 실체 및 그들사이의 호상관계를 간단히 말하고 일상적으로 보게 되는 물체에서 객체의 개념을 끌어내시오.**

실체는 현실세계의 물리적존재이다. 객체는 현실세계의 어떤 구체적인 물리적인 실체를 콤퓨터론리에로 옮긴것이다.

클라스는 동일한 형의 실체에 대응하는 모든 객체들의 추상과 공통적인 특징, 행위의 모임이다. 실례로 차는 클라스이며 빨간색의 차는 클라스의 객체이다.

**1.1.3. 객체는 어떤 속성을 가지는가? 상태와 행위란 무엇이며 그들사이에 어떤 관계가 있는가? 《학생》을 객체라고 할 때 이 객체에 대하여 상태와 행위를 정의하시오.**

객체는 정적속성과 동적속성을 가진다. 정적속성은 객체의 상태를 표시하는데 그것을 객체의 마당(령역)이라고 한다.

동적속성은 객체의 조작을 표시하는데 그것을 객체의 행위 또는 메쏘드라고 한다.

행위는 객체내부정보를 포함하는 상태를 객체내부에 매몰시키며 객체내부정보가 객체외부와 교차하는 안전한 조종과 접속을 제공한다.

《학생객체》의 상태에는 번호, 이름, 성별, 나이, 학급 등이 속하며 행위에는 학생번호수정, 학급조절, 학생기본정보인쇄 등이 속한다.

**1.1.4. 객체들사이에 어떤 관계가 있는가? 《학급》객체와 《학생》객체, 《학생》객체와 《대학생》객체는 서로 어떤 관계인가?**

객체들사이에는 포함, 계승, 련관의 3가지 관계가 있다.

《학급》객체와 《학생》객체는 련관관계이며 《학생》객체와 《대학생》객체는 계승관계이다.

**1.1.5. 《대학》과 《건설대학》은 계승관계인가 아닌가, 그 리유를 말하시오.**

건설대학은 클라스가 아니라 객체이다. 건설대학은 대학클라스의 객체이다.

따라서 계승관계가 아니다.

**1.1.6. 객체지향프로그람개발은 어떤 과정을 포함하는가? OOA모형은 어떤 5개의 층을 포함하는가? OOD모형은 OOA모형을 기초하여 어떤 작업을 인입하였는가?**

객체지향프로그람개발은 주로 객체지향의 분석(OOA), 객체지향의 설계(OOD), 객체지향의 실현(OOP) 및 그 이후의 검사, 보수 등의 과정을 포함한다.

OOA모형은 객체-클라스층, 정적속성층, 봉사층, 구조층, 주제층(subject)의 5개 층을 포함한다.

OOD모형은 OOA모형을 기초로 확장되였으며 대면관리, 업무관리, 자료관리의 3개 부분의 내용을 인입하였다.

**1.1.7. 객체지향프로그람설계방법은 어떤 우점을 가지는가?**

재리용가능성, 확장가능성, 관리가능성과 같은 3가지 우점을 가진다.

**1.1.8. JDK쏘프트웨어패키지를 리용하여 이 프로그람을 번역 및 실행하고 화면상에 《Welcome to Java World!》를 출력하는 Java Application을 작성하시오.**

원천프로그람 MyJavaExcerciseApplication.java

```
import java.io.*;
public class MyJavaExcerciseApplication
{
    public static void main(String args[ ])
    {
        System.out.println("Welcome to Java World!");
    }//end of main method
}//end of class
```

번역명령　javac MyJavaExcerciseApplication.java

실행명령　java MyJavaExcerciseApplication

**1.1.9. 열람기에서 《Welcome to Java Applet World!》문자렬정보를 현시하는 Java Applet를 작성하시오.**

원천프로그람 MyJavaExcerciseApplet.java

```
import java.awt.Graphics;
import java.applet.Applet;
public class MyJavaExcerciseApplet extends Applet
{
```

```
        public void paint(Graphics g)
        {
            g.drawString("Welcome to Java Applet World!", 10, 20);
        }// end of paint method
    }// end of class
```

**1.1.10.** HTML파일을 작성하고 문제 1.1.9에서 만든 Applet바이트코드를 거기에 삽입하시오. 그리고 WWW열람기를 리용하여 이 HTML파일이 규정하는 Web페지를 보시오.

원천프로그람 MyAppletInclude.html

```
<HTML>
    <BODY>
        <APPLET CODE="MyJavaExcerciseApplet.class" HEIGHT=200 WIDTH=300>
        </APPLET>
    </BODY>
</HTML>
```

**1.1.11.** 표식자객체 myLabel을 리용하여 《Java는 객체지향언어이다.》라는 문자렬정보를 출력하는 Java Applet를 작성하시오.

원천프로그람 AppletLabelOutput.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class AppletLabelOutput extends Applet
{
    Label myLabel;
    public void init( )
    {
        myLabel = new Label("Java는 객체지향언어이다.");
        add(myLabel);
    }
} //end of class
```

**1.1.12.** 사용자가 입력한 한행의 문자렬을 접수하고 3개행을 반복하여 출력하는 Java Application을 작성하시오.

원천프로그람 ApplicationInOut.java

```
import java.io.*;
public class ApplicationInOut
```

```
{
    public static void main(String args[ ])
    {
        String s=" ";
        System.out.print("please enter a string:");
        try
        {
            BufferedReader in = new
            BufferedReader(new InputStreamReader(System.in));
            s=in.readLine( );
        }catch(IOException e){ }
        System.out.println("You've entered string:");
        System.out.println(s);
        System.out.println(s);
        System.out.println(s);
    }
}
```

### 1.1.13. Java언어는 어떤 우점이 있는가?

java언어의 우점은 가동환경의 무관계성, 객체지향성, 안전성이며 다중처리를 지원하고 학습하기 쉬운것이다.

## 1.2. 보충문제

1.2.1. 객체지향설계의 주요사상은 무엇인가?

1.2.2. 왜 객체지향방법이 높은 수준의 정보밀봉을 실현하는데 유리한가?

1.2.3. 현실에서 클라스, 객체, 클라스의 계승, 포함, 련관실례를 드시오.

1.2.4. 《부문》, 《대학》, 《학장》, 《교원》의 4개 클라스가 있다. 매개 클라스에 대하여 마당과 행위를 설계하고 그것들사이의 관계를 말하시오.

1.2.5. 객체지향의 사상과 방법을 리용하여 자동출납기체계를 설계하시오.

체계에서 어떤 클라스들을 요구하는가? 클라스들은 어떤 속성을 가지며 그것들사이의 관계는 어떤가?

1.2.6. 바이트코드번역기란 무엇이며 즉시번역(just-in-time)이란 무엇인가?

1.2.7. Java프로그람의 기본구조를 말하시오. 클라스이름과 클라스를 보존하는 원천프로그람파일이름, 바이트코드파일이름들사이에 어떤 관계가 있는가?

하나의 Java원천프로그람을 번역하면 몇개의 바이트코드파일이 만들어지는가?

1.2.8. 다음 개념들의 차이점을 말하시오.

1) 하드웨어, 쏘프트웨어
2) 체계쏘프트웨어, 응용쏘프트웨어
3) 번역기, 해석기
4) 번역, 실행
5) 기계언어, 고급언어
6) Internet, WWW
7) HTTP, HTML
8) 원천코드, 목표코드(object code), 바이트코드
9) Application, Applet

# 제2장. Java언어기초

## 2.1. 련습

**2.1.1. Java프로그람구조를 간단히 말하시오. 주클라스를 어떻게 판단하며 다음 프로그람의 틀린곳을 어떻게 고쳐야 하는가? 이 프로그람의 원천코드는 어떤 이름의 파일로 보관해야 하는가?**

```
public class MyJavaClass
{
    public static void main(String[ ] args)
    {
        System.out.println("Java프로그람");
    }
    System.out.println("프로그람결속");
}
```

Java프로그람은 하나 또는 여러개의 클라스정의로 이루어진다. 그 중에 하나는 반드시 주클라스여야 한다. Java Application프로그람의 주클라스는 main( )메쏘드를 포함하고있는 클라스이다. Java Applet프로그람의 주클라스는 체계클라스 Applet의 하위클라스이다.

주클라스의 클라스이름은 이 Java원천프로그람의 파일이름이다. 문제의 원천코드에서는 오직 한개의 클라스 MyJavaClass만을 정의하였다. 이 클라스가 주클라스이다. 따라서 이 프로그람의 원천코드는 MyJavaClass.java로 보관하여야 한다. 원천코드의 결함은 명령을 클라스정의 밖의 임의의곳에 독립적으로 놓은데 있다. 정확히 쓰면 다음과 같다.

```
public class MyJavaClass
{
    public static void main(String args[ ])
    {
        System.out.println("Java프로그람");
        System.out.println("프로그람결속");
    }
}
```

**2.1.2. Java에는 어떤 기본자료형이 있는가? int형이 표시할수 있는 최대, 최소값범위를 말하시오.**

Java의 기본자료형에는 론리형, 바이트형, 문자형, 짧은 옹근수형, 옹근수형, 긴 옹근수형, 류동소수점형, 배정확도형이 있다.

int형이 표시할수 있는 최대값은 2147483647이며 최소값은 -2147483648이다.

**2.1.3. Java에서 문자는 어떤 코드작성방법을 리용하며 어떤 특징을 가지는가? 자주 리용하는 전의부를 5개 쓰시오.**

Java에서 문자는 Unicode코드작성방법을 리용한다. 그 특징은 포함된 정보량이 최대이고 동양, 서양문자를 모두 하나의 Unicode문자로 표시할수 있으며 ASCII코드작성과 같이 1개 문자로 서양문자를 표현하고 2개 문자로 동양문자를 표시하지 않아도 된다는것이다.

자주 리용하는 전의부는 다음과 같다.

'\b'→ 후퇴하기, '\n'→ 행바꾸기, '\r'→ 되돌이, '\"'→ 쌍인용괄호, '\\'→ 역사선

**2.1.4. Java에서 식별부를 어떻게 규정하고있는가? 아래의 식별부에서 어느것이 옳고 틀린것은 어느것인가?**

1) MyGame  2) _isHers  3) 2JavaProgram  4) Java-Visual-Machine  5) _$ abc

Java에서 식별부는 자모, 수자, 밑선, $기호로 이루어지며 반드시 자모, 밑선, $기호로 시작된다.

문제에서 1), 2), 5)는 정확한 식별부이다. 3)은 수자로 시작되고 4)는 가로선을 가지고있으므로 틀린 식별부이다.

**2.1.5. 상수와 변수란 무엇이며 문자변수와 문자렬상수는 어떤 차이점을 가지는가?**

상수는 수값 혹은 자료가 이미 만들어져있으며 프로그람의 전체 실행과정에 변할수 없는것을 말한다.

변수는 수값 혹은 자료가 프로그람의 실행과정에서 변할수 있는것을 말한다.

아래의 명령은 하나의 문자변수를 정의한다.

```
char ch ='a';
```

변수 ch의 초기값은 'a'로 되며 만일 값주기명령 ch='b'를 실행하면 ch의 값은 'b'로 변한다.

문자렬상수는 쌍인용괄호를 리용한 문자렬이다. 실례로 "a".

**2.1.6. 강제형변환이란 무엇이며 어떤 조건에서 강제형변환을 리용해야 하는가?**

내부기억기를 비교적 많이 차지하는 자료형을 내부기억기를 적게 차지하는 자료형으로 변환할 때 명백히 형변환선언을 해야 한다. 그 형식은 다음과 같다.

(자료형)변수이름 혹은 표현식

이것을 강제형변환이라고 한다.

**2.1.7. Java에는 어떤 산수연산자, 관계연산자, 론리연산자, 비트연산자, 값주기연산자들이 있는가? 1항연산자, 2항연산자, 3항연산자의 실례를 말하시오.**

1) Java의 산수연산자

1항연산자  +, -, *, /, %

2항연산자  ++, --, -

2) 관계연산자 ==, !=, >, <, >=, <=
3) 론리연산자 &&, ||, !
4) 비트연산자
   비트론리연산자 &, |, ^, ~,
   비트옮김연산자 >>, <<, >>>, <<<
5) 값주기연산자 +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=, <<<=
6) 3항연산자 ?, :

**2.1.8. 다음 식의 결과를 쓰시오. a=3, b=-5, f=true일 때**

1) --a%b++ 2) (a>=1&&a<=12?a:b) 3) f^(a>b) 4) (--a)<<a

1) --a % b++ = 2
2) (a >= 1 && a <= 12 ? a : b) = 3
3) f ^ (a > b) = false
4) (--a) << a = 8

```
원천프로그람
public class ch2_e2_8
{
    public static void main(String args[ ])
    {
        int a = 3, b = -5;
        boolean f = true;
        System.out.println("--a % b++ = " + (--a % b++));
        a=3; b=-5; //왜 이 행이 있어야 하며 없으면 실행결과는 어떤가?
        System.out.println("(a >= 1 && a <= 12 ? a : b) = " + (a >= 1 && a <= 12 ? a : b));
        System.out.println("f ^ (a > b) = " + (f ^ (a > b)));
        System.out.println("(--a) << a = " + ((--a) << a));
    }
}
```

**2.1.9. 사용자가 입력한 류동소수점수를 접수하여 그의 소수부와 옹근수부를 각각 출력하는 문자대면의 Java Application프로그람을 작성하시오.**

```
원천프로그람 ch2_e2_9.java
import java.io.*;
public class ch2_e2_9
```

```
{
  public static void main(String args[ ])
  {
    String s;
    double d;
    int i;
    boolean b = false;
    do{
      try{
        System.out.println(" 하나의 류동소수점수를 입력하시오.");
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        s = br.readLine( ); //문자렬방식으로 읽어들이기
        i = s.indexOf('.'); //소수점의 위치찾기
        d = Double.parseDouble(s); //문자렬을 류동소수점수로 변환하기
        System.out.println(d + "옹근수부는 다음과 같다: " + (long)d);
        if (i == -1) //만일 소수점이 없다면 소수부는 없다.
          System.out.println(d + "소수부는 다음과 같다: 0.0");
        else //소수점이 있다면 소수점이후의 문자렬을 잘라버리고 류동소수점수로 합성한다.
          System.out.println(d + "소수부는 다음과 같다 :"
                      + Double.parseDouble(((s.charAt(0) == '-') ? "-" : "  ")
                      + "0." + s.substring(i + 1, s.length( ))));
        b = false;
      }
      catch(NumberFormatException nfe)
      {
        System.out.println("류동소수점수형식의 입력이 틀린다. \n");
        b = true;
      }
      catch(IOException ioe)
      {
        b = false;
      }
    }while(b);    //류동소수점수형식이 틀릴 때 다시 입력한다.
  } // end of main
}//end of class
```

2.1.10. 사용자가 입력한 10개의 옹근수를 접수하여 그의 최대, 최소값을 비교하고 출력하는 문자대면의 Java Application프로그람을 작성하시오.

```
원천프로그람 ch2_e2_10.java
import java.io.*;
public class ch2_e2_10
{
  public static void main(String args[ ])
  {
    int max = 0, min = 0, value = 0;
    for(int i = 1; i <= 10; i++)
    {
      try{
        System.out.println("입력" + i + " 옹근수:");
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String s = br.readLine( );
        value = Integer.parseInt(s);
        if(i == 1)
        {
          max = min = value;
        }
        else
        {
          max = value > max ? value : max;
          min = value < min ? value : min;
        }
      }
      catch(NumberFormatException enf)
      {
        System.out.println("옹근수형식의 입력이 틀리면 다시 입력한다. \n");
        i --;
      }
      catch(IOException ioe)
      {
         System.err.println(ioe.toString( ));
         System.out.println("일반적으로 입출력이 틀리면 프로그람을 중지한다. ");
         System.exit(0);
```

```
      }
    } //end of for
    System.out.println("최대값: " +  max);
    System.out.println("최소값: " + min);
  } //end of method
} //end of class
```

**2.1.11. 사용자가 입력한 문자를 접수하여 영어자모순으로 최소인 문자를 비교하고 출력하며 《#》이 입력되면 프로그람이 끝나는 문자대면의 Java Application프로그람을 작성하시오.**

```
원천프로그람 ch2_e2_11.java
import java.io.*;
public class ch2_e2_11
{
  public static void main(String args[ ])
  {
    char ch,min;
    try{
      System.out.println("한개 문자를 입력하며 '#'으로 결속한다.");
      ch = (char)System.in.read( );
      min = ch;
      System.in.skip(2);
      while(ch !=  '#')
      {
        ch = (char)System.in.read( );
        min = (ch < min && ch != '#') ? ch : min;
        System.in.skip(2); //입력건 건너뛰기
      }
      System.out.println("\n입력문자에서 영어자모순으로 최소인것은: " + min);
    }
    catch(IOException ioe)
    {
    System.err.println(ioe.toString( ));
    }
  }
}
```

**2.1.12. 구조화프로그람설계에 어떤 기본흐름명령이 있는가? 그것이 Java의 어떤 명령에 각각 대응되는가?**

구조화프로그람설계에는 순서, 분기, 순환의 3가지 기본흐름명령이 있다. Java의 분기명령에는 if명령과 switch명령이 속하며 순환명령에는 while명령과 do-while명령, for명령이 속한다. 그리고 순서명령에는 객체와 변수정의명령, 값주기명령, 메쏘드리용명령, 메쏘드선택리용명령 등이 속한다.

**2.1.13. 3개 옹근수를 입력하여 최소값을 구하는 프로그람을 작성하시오.**

원천프로그람 ch2_e2_13.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch2_e2_13 extends Applet implements ActionListener
{
  Label result;
  TextField in1, in2, in3;
  Button btn;
  int a = 0, b = 0, c = 0, min = 0;
  public void init( )
  {
    result = new Label("비교하려는 3개의 옹근수를 먼저 입력하시오.");
    in1 = new TextField(5);
    in2 = new TextField(5);
    in3 = new TextField(5);
    btn = new Button("비교");
    add(in1);
    add(in2);
    add(in3);
    add(btn);
    add(result);
    btn.addActionListener(this);
  }
  public void actionPerformed(ActionEvent e)
  {
    a = Integer.parseInt(in1.getText( ));
    b = Integer.parseInt(in2.getText( ));
```

```
c =  Integer.parseInt(in3.getText( ));
if(a < b)
  if(a < c)
    min = a;
  else
    min = c;
  else
    if(b < c)
      min = b;
    else
      min = c;
result.setText("세수가운데서 최소값은:" + min);
 }
}
```

**2.1.14. 사용자가 입력한 1～12의 옹근수(만일 입력한 자료가 이 조건을 만족하지 않으면 사용자가 다시 입력할것을 요구한다.)를 접수하고 switch명령을 리용하여 매 월에 대응하는 월의 일수를 출력하는 Java프로그람을 작성하시오.**

원천프로그람 ch2_e2_14.java

```
import java.io.*;
public class ch2_e2_14
{
 public static void main(String args[ ])
 {
   int i = 0;
   do
   {
     try
     {
          System.out.println("1～12의 하나의 옹근수를 입력한다.");
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      String s = br.readLine( );
      i = Integer.parseInt(s);
     }
    catch(NumberFormatException nfe)
    {
     System.out.println("입력형식이 틀린다.");
```

```
     i = -1;
    }
    catch(IOException ioe)
   {
     System.err.println(ioe.toString( ));
     System.exit(0);
   }
  }while(i < 1 | | i > 12);
  switch(i)
 {
    case 1:
         System.out.println("1월은 31일간");
         break;
    case 2:
         System.out.println("2월은 28일 혹은 29일");
         break;
    case 3:
         System.out.println("3월은 31일간");
         break;
    case 4:
         System.out.println("4월은 30일");
         break;
    case 5:
         System.out.println("5월은 31일");
         break;
    case 6:
         System.out.println("6월은 30일");
         break;
    case 7:
         System.out.println("7월은 31일");
         break;
    case 8:
         System.out.println("8월은 31일");
         break;
    case 9:
         System.out.println("9월은 30일");
         break;
```

```
    case 10:
        System.out.println("10월은 31일");
        break;
    case 11:
        System.out.println("11월은 30일");
        break;
    case 12:
        System.out.println("12월은 31일");
        break;
   }
  } //end of main method
} //end of class
```

**2.1.15. 순환에서 break, continue와 return명령을 리용하면 어떤 서로 다른 효과가 있는가?**

순환에서 break를 리용하면 순환이 정지되며 흐름은 break명령이 있는곳으로부터 break명령이 있는 순환밖의 첫번째 명령까지 건너뛰여 계속 실행한다.

순환에서 continue명령을 리용하면 먼저 본순환을 결속하고 흐름은 continue명령이 있는 순환의 첫번째 명령까지 건너뛰여 계속 실행한다.

순환에서 return명령을 리용하면 현재 메쏘드선택리용을 중지하며 동시에 순환을 정지한다. 또한 명령을 선택리용한 다음의 명령까지 되돌아가서 실행하게 한다.

**2.1.16. 사용자가 입력한 2개의 자료를 웃한계, 아래한계로 하여 그사이에 있는 모든 씨수를 출력하는 도형대면의 Java Applet프로그람을 작성하시오.**

```
원천프로그람 ch2_e2_16.java
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch2_e2_16 extends Applet implements ActionListener
{
  Label prompt;
  TextField ceiling,floor;
  TextArea result;
  Button btn;
  int max = 0, min = 0, temp = 0;
  String resultString = "이 범위안의 씨수는 다음과 같다.\n";
  public void init( )
```

```
{
  prompt = new Label("출력하려는 씨수의 웃한계, 아래한계 범위를 입력하십시오:");
  ceiling = new TextField(5);
  floor = new TextField(5);
  result = new TextArea(8,30);
  btn = new Button("출력");
  add(prompt);
  add(ceiling);
  add(floor);
  add(btn);
  add(result);
  btn.addActionListener(this);
}
public void actionPerformed(ActionEvent e)
{
  try
  {
    max = Integer.parseInt(ceiling.getText( ));
    min = Integer.parseInt(floor.getText( ));
    if(max < min)
    {
      temp = max;
      max = min;
      min = temp;
    }
    for(int i = min; i <= max; i++)
    {
      boolean flag = true;
      if(i == 2 || i == 3)
      {
        resultString = resultString+Integer.toString(i) + "\n";
        continue;
      }
      if(i % 2 == 0)
        continue;
      else
        for(int j = 3; j < i; j += 2)
```

```
            {
              if(i % j == 0)
                {flag = false;
                break;}
        }
        if(flag)
          resultString = resultString+Integer.toString(i) + "\n";
        }//for(i)
        result.setText(resultString);
        resultString = "이 범위안의 씨수는 다음과 같다. \n";
      }
      catch(NumberFormatException nfe)
      {
        result.setText("형식이 틀리면 정의옹근수를 입력한다.");
      }
    } //end of actionPerformed( ) methed
  } //class of class
```

## 2.1.17. 사용자가 입력한 수의 인수가운데서 씨수를 출력하는 프로그람을 작성하시오.

원천프로그람 ch2_e2_17.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch2_e2_17 extends Applet implements ActionListener
{
  Label prompt;
  TextField input;
  TextArea result;
  Button btn;
  long value=0;
  int count=0;
  String resultString="이 수의 인수가운데서 씨수들은 다음과 같다.\n";
  public void init( )
  {
    prompt = new Label("씨수를 구하려는 정의옹근수를 입력하시오.");
    input = new TextField(5);
    result = new TextArea(8,30);
```

```
    btn = new Button("출력");
    add(prompt);
    add(input);
    add(btn);
    add(result);
    btn.addActionListener(this);
  }
  public void actionPerformed(ActionEvent e)
  {
    try
    {
      value=Long.parseLong(input.getText( ));
      if(value%2==0)
      {
        resultString=resultString+"2\t";
        count++;
        value/=2;
      }
      if(value%3==0)
      {
        resultString=resultString+"3\t";
        count++;
        value/=3;
      }
      for(int i=5; i<=value; i+=2)
      {
        if(value%i !=0)
          continue;
        else
        {
          boolean flag = true;
          if (i % 2==0)
            continue;
          else
            for(int j=3; j<i; j+=2)
            {
              if(i%j==0)
```

```
            {flag = false;
            break;}
        }
        if(flag)
        {
          resultString =resultString+ Integer. toString (i);
          if( ++ count <4)
            resultString = resultString + "\t";
          else
          {
            count=0;
            resultString = resultString + "\n";
          }
        } //if(flag)
      value /= i;
      } //else
    } //for(i)
    result.setText(resultString);
    resultString = "이 수의 인수가운데서 씨수들은 다음과 같다.\n";
  }
  catch(NumberFormatException nfe)
  {
    result.setText("형식이 틀리면 정의옹근수를 입력한다.");
  }
 } //end of actionPerformed( )
} //end of class
```

**2.1.18. 배렬이란 무엇이며 어떤 특징이 있는가? Java에서 배렬을 만들자면 어떤 단계를 거쳐야 하며 배렬의 원소에 어떻게 접근하는가? 배렬원소의 첨수와 배렬의 길이는 어떤 관계가 있는가?**

배렬은 같은 자료형의 원소들을 일정한 순서에 따라 선형배렬하여 묶어놓은것을 말한다.

Java프로그람에서 배렬을 만들자면 배렬선언, 배렬공간창조, 배렬원소창조 및 초기화의 3단계를 거쳐야 한다.

배렬이름과 아래첨수를 리용하여 배렬원소에 접근할수 있다. 만일 배렬에 N개 원소가 있다면 배렬의 길이는 N이고 배렬원소의 아래첨수는 0부터 N-1까지다.

**2.1.19. 옹근수배렬의 최대값, 최소값, 평균값, 배렬의 모든 원소의 합을 구하는 프로그람을 작성하시오.**

원천프로그람 ch2_e2_19.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch2_e2_19 extends Applet implements ActionListener
{
  final int ARRAY_LENGTH = 10;
  final String labelPrompt[ ] = {"최대값: ", "최소값: ", "총합: ", "평균값: "};
  int myArray[ ] = new int[ARRAY_LENGTH];
  int count = 0,max = 0,min = 0,sum = 0;
  double avg = 0.0;
  TextField inputTfd = new TextField(10);
  Label inputLbl = new Label("자료를 입력하시오:");
  Label maxLabel = new Label(labelPrompt[0]+" ");
  Label minLabel = new Label(labelPrompt[1]+" ");
  Label sumLabel = new Label(labelPrompt[2]+" ");
  Label avgLabel = new Label(labelPrompt[3]+" ");
  public void init( )
  {
    for(int i = 0; i < ARRAY_LENGTH; i++)
    {
      myArray[i] = 0;
    }
    add(inputLbl);
    add(inputTfd);
    add(maxLabel);
    add(minLabel);
    add(sumLabel);
    add(avgLabel);
    inputTfd.addActionListener(this);
  }
  public void actionPerformed(ActionEvent ae)
  {
    TextField temp = (TextField)(ae.getSource( ));
```

```
      if(temp == inputTfd)
      {
        try
        {
          int value = Integer.parseInt(temp.getText( ));
          if(count == 0)
          {
            max = value;
            min = value;
            sum = value;
            avg = value;
          }
          else
          {
            max = Math.max(value,max);
            min = Math.min(value,min);
            sum = (count < ARRAY_LENGTH) ? sum + value :
                                sum - myArray[count % ARRAY_LENGTH] + value;
            avg = ((double)(sum)) / (count < ARRAY_LENGTH ? count + 1 : ARRAY_LENGTH);
          }
          myArray[count%ARRAY_LENGTH] = value;
          count++;
          maxLabel.setText(labelPrompt[0] + max);
          minLabel.setText(labelPrompt[1] + min);
          sumLabel.setText(labelPrompt[2] + sum);
          avgLabel.setText(labelPrompt[3] + avg);
          inputTfd.setText(" ");
        }
        catch(NumberFormatException nfe)
        {
          inputTfd.setText("형식이 틀린다.");
        }
      }
      else
      {
        showStatus("사건" + ae.toString( ) + "처리조작을 정의하지 않았다.");
      }
    }
  }
```

**2.1.20. 벡토르와 배렬이 어떻게 다른가? 그것들은 각각 어떤 경우에 리용하는것이 좋은가?**

벡토르는 배렬의 순서기억과 류사한 자료구조이다. 다른 점은 벡토르의 매개 원소는 모두 객체이고 벡토르에서는 자주 리용하는 일부 자료처리기능을 매몰하였다는것이다. 또한 벡토르는 서로 다른 형들의 원소가 공존하는 가변길이의 배렬을 리용한다.

아래의 조건에서는 벡토르를 리용하는것이 좋다.

- 처리하려는 객체수가 정확하지 않고 순서렬의 원소들이 모두 객체이거나 객체로 표시할수 있다.
- 서로 다른 클라스의 객체를 하나의 자료순서렬로 만들어야 한다.
- 복잡한 객체순서렬에서 원소의 삽입과 삭제를 하여야 한다.
- 순서렬에서 객체의 위치를 정하거나 혹은 다른 탐색조작을 해야 한다.
- 서로 다른 클라스들사이에 많은 자료를 전달하여야 한다.

Vector클라스의 방법이 배렬에 비하여 많지만 이 클라스를 리용하는것 역시 일정한 제한성이 있다. 실례로 그 가운데 객체는 간단한 자료형이 될수 없다.

아래의 조건에서는 배렬을 리용하는것이 비교적 좋다.

- 순서렬의 원소들이 모두 간단한 자료형의 자료이다.
- 순서렬의 원소개수가 상대적으로 고정되고 삽입, 삭제, 찾기조작이 비교적 적다.

**2.1.21. 200개의 전화카드객체를 만드는 Applet프로그람을 작성하시오. 프로그람은 200개의 전화카드번호를 자동적으로 만들어내야 하며 사용자에 의하여 암호(암호를 입력하는 본문칸은 *문자를 리용한다.)와 금액이 입력되며 받는 사람번호와 부가비는 200과 0.1로 고정한다. 매번 200개의 전화카드객체를 만든 다음 그와 관계되는 정보를 출력한다. 프로그람은 임의의 개수의 객체를 만들수 있으며 그것을 금액이 높은 순서로 하나의 벡토르객체에 배렬하시오.**

원천프로그람 ch2_e2_21.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class ch2_e2_21 extends Applet implements ActionListener
{
  final long CONNECT_NUMBER = 200;
  final double CONNECT_FEE = 0.5;
  Label cardNumberLbl = new Label( );
  Label balanceLbl = new Label("나머지금액");
```

```
Label passwordLbl = new Label("암호");
TextField balanceTfd = new TextField(8);
TextField passwordTfd = new TextField(8);
Button createCard = new Button("만들기");
Vector PhoneCardVector = new Vector( );
public void init( )
{
  balanceTfd.setText(" ");
  passwordTfd.setText(" ");
  cardNumberLbl.setText("전화카드번호" + PhoneCard200.cardNumber200);
  add(cardNumberLbl);
  add(balanceLbl);
  add(balanceTfd);
  add(passwordLbl);
  add(passwordTfd);
  add(createCard);
  passwordTfd.setEchoChar('*');
  createCard.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
  PhoneCard200 phoneCard;
  if(ae.getSource( ) == createCard)
  {
    try
    {
      //전화카드만들기
      phoneCard = new PhoneCard200(
      Double.parseDouble(balanceTfd.getText( )), passwordTfd.getText( ),
      CONNECT_NUMBER, CONNECT_FEE);
      //높은 순서로 삽입
      boolean added = false;
      for(int i = 0; i < PhoneCardVector.size( ); i++)
      {
        if(phoneCard.getBalance() < ((PhoneCard200)(PhoneCardVector.get(i))).getBalance())
        {
          PhoneCardVector.insertElementAt(phoneCard, i);
```

```
                added = true;
                break;
              }
            }
            //만일 첫번째 카드 혹은 남은 금액이 최대인 카드라면 벡토르의 마지막에 추가
            if(!added)
              PhoneCardVector.add(phoneCard);
            cardNumberLbl.setText("전화카드번호:" + PhoneCard200.cardNumber200);
            balanceTfd.setText(" ");
            passwordTfd.setText(" ");
            repaint( );
          }
          catch(NumberFormatException nfe)
          {
            showStatus("입력한 남은 금액형식이 틀리면 다시 입력한다.");
            balanceTfd.setText(" ");
            passwordTfd.setText(" ");
          }
        }
        else
        {
          showStatus("사건" + ae.toString( ) + "처리조작을 정의하지 않았다.");
          balanceTfd.setText(" ");
          passwordTfd.setText(" ");
        }
      }
      public void paint(Graphics g)
      {
        for(int i = 0; i < PhoneCardVector.size( ); i++)
        {
          g.drawString(PhoneCardVector.get(i).toString( ), 10, 70 + 20 * i);
        }
      }
    }
    class PhoneCard200
    {
      static long cardNumber200;
```

```
long cardNumber;
double balance;
String password;
Long connectNumber;
Double connectFee;
static
{
  cardNumber200 = 100001;
}
public PhoneCard200(double b, String pwd, long cNum, double cFee)
{
  cardNumber = cardNumber200++;
  balance = b;
  password = new String(pwd);
  connectNumber = cNum;
  connectFee = cFee;
}
public double getBalance( )
{
  return balance;
}
public String toString( )
{
  String s = "카드번호" + cardNumber + "남은 금액" + balance + "암호"
       + password + "련결번호" + connectNumber + "련결비용" + connectFee;
  return s;
}
}
```

**2.1.22. 문제 2.1.21에 기초하여 탐색기능을 추가하고 사용자가 하나의 카드번호를 입력하면 프로그람은 대응하는 전화카드의 금액을 출력하도록 하며 만일 찾지 못하면 대응하는 제시정보를 출력하시오.**

원천프로그람 ch2_e2_22.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
```

```
public class ch2_e2_22 extends Applet implements ActionListener
{
  final long CONNECT_NUMBER = 200;
  final double CONNECT_FEE = 0.5;
  Label cardNumberLbl = new Label( );
  Label balanceLbl = new Label("나머지금액");
  Label passwordLbl = new Label("암호");
  TextField balanceTfd = new TextField(8);
  TextField passwordTfd = new TextField(8);
  TextField searchIdxTfd = new TextField(8);
  Button createCard = new Button("만들기");
  Button searchCard = new Button("탐색");
  Label searchResult = new Label("탐색결과:" + " ");
  Vector phoneCardVector = new Vector( );
  public void init( )
  {
    balanceTfd.setText(" ");
    passwordTfd.setText(" ");
    searchIdxTfd.setText(" ");
    cardNumberLbl.setText("전화카드번호 " + PhoneCard200.cardNumber200);
    add(cardNumberLbl);
    add(balanceLbl);
    add(balanceTfd);
    add(passwordLbl);
    add(passwordTfd);
    add(createCard);
    add(searchIdxTfd);
    add(searchCard);
    add(searchResult);
    passwordTfd.setEchoChar('*');
    searchResult.setAlignment(Label.LEFT);
    createCard.addActionListener(this);
    searchCard.addActionListener(this);
  }
  public void actionPerformed(ActionEvent ae)
  {
    PhoneCard200 phoneCard;
```

```
if(ae.getSource( ) == createCard)
{
  try
  {
    //전화카드만들기
    phoneCard = new PhoneCard200(
    Double.parseDouble(balanceTfd.getText( )),
    passwordTfd.getText( ), CONNECT_NUMBER, CONNECT_FEE);
    //높은 순서로 삽입
    boolean added = false;
    for(int i = 0; i < phoneCardVector.size( ); i++)
    {
      if(phoneCard.getBalance( ) <
      ((PhoneCard200)(phoneCardVector.get(i))).getBalance( ))
      {
        phoneCardVector.insertElementAt(phoneCard,i);
        added = true;
        break;
      }
    }
    //만일 첫번째 카드 혹은 남은 금액이 최대인 카드라면 벡토르의 제일 마지막에 추가
    if(!added)
      phoneCardVector.add(phoneCard);
    cardNumberLbl.setText("전화카드번호" + PhoneCard200.cardNumber200);
    balanceTfd.setText(" ");
    passwordTfd.setText(" ");
    repaint( );
  }
  catch(NumberFormatException nfe)
  {
    showStatus("입력한 남은 금액형식이 틀리면 다시 입력한다.");
    balanceTfd.setText(" ");
    passwordTfd.setText(" ");
  }
}
else if(ae.getSource( ) == searchCard)
{
```

```
        try
        {
          Object obj;
          boolean found=false;
          long searchIndex=Long.parseLong(
          searchIdxTfd.getText( ));
          for(Enumeration e=phoneCardVector.elements( );
          e.hasMoreElements( );)
          {
            obj=e.nextElement( );
            if(obj instanceof PhoneCard200)
            {
              if(((PhoneCard200)obj).getCardNumber( ) == searchIndex)
              {
                searchResult.setText("탐색결과: " + ((PhoneCard200)obj).toString( ));
                found=true;
                break;
              }
            }
          }
          if(! found)
          {
            searchResult.setText("찾지못함");
            searchIdxTfd.setText(" ");
          }
        }
        catch(NumberFormatException nfe)
        {
          showStatus("탐색령역에서 입력한 카드번호형식이 틀린다.");
          searchIdxTfd.setText(" ");
          searchResult.setText("탐색결과:");
        }
      }
      else
      {
        showStatus("사건" + ae.toString( ) + "처리조작이 정의되지 않았다.");
        balanceTfd.setText(" ");
```

```
            passwordTfd.setText(" ");
        }
    }
    public void paint(Graphics g)
    {
        for(int i = 0; i < phoneCardVector.size( ); i++)
        {
            g.drawString(phoneCardVector.get(i).toString( ), 10, 150 + 20 * i);
        }
    }
}
class PhoneCard200
{
    static long cardNumber200;
    long cardNumber;
    double balance;
    String password;
    Long connectNumber;
    double connectFee;
    static
    {
        cardNumber200 = 100001;
    }
    public PhoneCard200(double b,String pwd,long cNum,double cFee)
    {
        cardNumber = cardNumber200++;
        balance = b;
        password = new String(pwd);
        connectNumber = cNum;
        connectFee = cFee;
    }
    public long getCardNumber( )
    {
        return cardNumber;
    }
    public double getBalance( )
    {
```

```
      return balance;
    }
    public String toString( )
    {
      String s = "카드번호: " + cardNumber + "남은 금액: " + balance
         + "암호번호: " + password + "련결번호: " + connectNumber
         + "련결비용: " + connectFee;
      return s;
    }
  }
```

**2.1.23. 문자렬이란 무엇이며 Java에서 문자렬은 어떤 클라스들로 갈라지는가?**

문자렬은 문자의 순서렬이다. Java프로그람에는 문자렬상수를 보관하는 String클라스와 문자렬변수를 보관하는 StringBuffer클라스가 있다.

**2.1.24. 사용자가 입력한 하나의 문자렬과 문자를 접수하여 문자렬에서 지정한 모든 문자를 삭제하고 출력하는 Applet프로그람을 작성하시오.**

```
원천프로그람 ch2_e2_24.java
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch2_e2_24 extends Applet implements ActionListener
{
  String originalString, modifiedString;
  int delChar;
  Label orgStringLbl = new Label("본래 문자렬:");
  Label delCharLbl = new Label("삭제하려는 문자:");
  TextField orgStringTfd = new TextField(20);
  TextField delCharTfd = new TextField(1);
  Button modifyBtn = new Button("삭제된 문자렬");
  public void init( )
  {
    add(orgStringLbl);
    add(orgStringTfd);
    add(delCharLbl);
    add(delCharTfd);
    add(modifyBtn);
```

```
    orgStringTfd.setText(" ");
    delCharTfd.setText(" ");
    originalString = " ";
    modifiedString = " ";
    modifyBtn.addActionListener(this);
  }
  public void paint(Graphics g)
  {
    g.drawString(modifiedString, 10, 150);
  }
  public void actionPerformed(ActionEvent ae)
  {
    if(ae.getSource( ) == modifyBtn)
    {
      originalString = orgStringTfd.getText( );
      delChar = (int)(delCharTfd.getText( ).charAt(0));
      modifiedString = "";
      int i = 0, j = 0;
      while((j = originalString.indexOf(delChar,i)) != -1)
      {
        System.out.println(i + "," + j + "," + (j - i));
        System.out.println(originalString);
        modifiedString = modifiedString
        +originalString.substring(i,j);
        System.out.println(modifiedString);
        i = j + 1;
      }
      modifiedString = modifiedString
      + originalString.substring(i,originalString.length( ));
      repaint( );
    }
    else
    {
      showStatus(" " + ae.toString( ) + "처리조작은 정의되지 않았습니다.");
      orgStringTfd.setText("");
      delCharTfd.setText("");
      originalString = " ";
```

```
            modifiedString = " ";
          }
        }
      }
```

2.1.25. 하나의 문자렬이 회문(palindrome)인가 아닌가를 판단하는 프로그람을 작성하시오.

```
원천프로그람 ch2_e2_25.java
import java.io.*;
public class ch2_e2_25
{
  public static void main(String args[ ])
  {
    try
    {
      System.out.println("한개 문자렬을 입력하시오:");
      BufferedReader br = new BufferedReader(
      new InputStreamReader(System.in));
      String str = br.readLine( );
      if(isReversable(str))
        System.out.println("이 문자렬은 회문이다.\n");
      else
        System.out.println("이 문자렬은 회문이 아니다.\n");
    }
    catch(IOException ioe)
    {
      System.out.println(ioe.toString( ));
    }
  }
  public static boolean isReversable(String s)
  {
    for(int i = 0, j = s.length( ) - 1; i < j; i++, j--)
    {
      if(s.charAt(i) != s.charAt(j))
      return false;
    }
    return true;
  }
}
```

**2.1.26.** String**클라스의** concat( )**메쏘드와** StringBuffer**클라스의** append( )**메쏘드는 두개의 문자렬을 련결할수 있다. 그것들사이의 다른 점은 무엇인가?**

String클라스의 concat( )메쏘드는 련결후 긴 문자렬을 되돌리며 String클라스자체는 변화되지 않는다.

StringBuffer클라스의 append( )메쏘드는 StringBuffer객체뒤에 하나의 문자렬을 추가하며 본래 있던 문자렬을 변화시킨다.

## 2.2. 보충문제

2.2.1. 예약어 void는 어떤 의미를 가지는가? 그것이 주로 어디에 리용되는가?

2.2.2. 아래의 자료에서 Java자료형을 지적하시오.

1) 23　　2) 315.08　　3) "315"　　4) "false"

5) false　　6) "9"　　7) "9"

2.2.3. 아래의 Java식별부가 옳은가를 말하시오.

1) int　　2) Int　　3) 28Elcamino　　4) 7!@#$!@#

5) Boolean　　6) public　　7) Private　　8) little brother

2.2.4. 마당과 메쏘드를 정의한 아래의 명령이 맞는가를 판단하고 틀린것을 고치시오.

1) public Boolean isTrue;

2) public boolean isTrue;

3) public boolean isTrue = "true";

4) public boolean is True;

5) string S;

6) public String Boolean;

7) public boolean isTrue = 0;

8) public int sum = 0.0

9) public double d = 9;

10) public String s = shabby;

11) public String boolean( );

12) public static void Main(String[ ] args);

2.2.5. boolean, int, double, String 등 형변수와 객체의 기정초기값을 쓰시오.

2.2.6. 프로그람이 8자리 유효수자까지 정확하게 계산할것을 요구한다면 어떤 자료형을 선택하여 리용하여야 하는가?

2.2.7. 아래의 요구에 근거하여 변수정의명령을 쓰시오.

1) 변수이름이 myBoolean이고 초기값이 참인 비공개론리형변수

2) 변수이름이 myString이고 초기값이 yes인 공개문자렬변수

3) 변수이름이 myDouble이고 초기값이 없는 비공개배정밀도변수

2.2.8. 론리형변수 B1, B2의 값이 참이고 B3, B4의 값이 거짓일 때 다음식들의 값을 말하시오.

1) (!B3 ^ B1) & (B2 ^ !B2)　　2) B1 ^ B3 | B2

3) B1 ^ B2 & B3　　4) !B1 & !B2

5) B1 & B2 & B3 & B4　　6) B1 | B2&B3

7) B1 || B3 && B4　　　　8) !B1 && !B3

2.2.9. 아래식의 값을 쓰시오.

1) 5 / 4　　　　2) 5.0 / 4
3) 5.0 / 4.0　　　　4) 5 % 4
5) 3 + 4 == 2 * 5　　　　6) 4 + 5 % 3 != 8 - 3
7) 4 + 5 / 6 >= 10 % 2　　　　8) 7 % 2 * 2 / 4 > 0

2.2.10. 변수 X, Y, Z의 값이 각각 3, 4, 5이다. 아래의 연산을 진행한 후 X, Y, Z의 값을 말하시오.

1) X = Y + Z++　　　　2) X *= Y++ - ++Z　　　　3) X = ++Y - Z++
4) X /= --Y – Z　　　　5) X = X++　　　　6) X += Y++ * Z

2.2.11. 아래의 연산자를 연산순서가 낮아지는 순서로 정렬하시오.

+, −, *, ++, ( ), /, %, <, ==

2.2.12. 아래의 요구를 만족하는 Java명령을 만드시오.

1) 만일 변수 B가 참이면 《참》을 인쇄하고 그렇지 않으면 《거짓》을 인쇄하시오.

2) 만일 변수 B1, B2이 다 참이면 《둘다 참》을 인쇄하고 그렇지 않으면 《거짓》을 인쇄하시오.

2.2.13. 아래의 Java명령에서 B1, B2이 어떤 값일 때 이 명령을 실행하여 《참》을 인쇄할수 있는가? 이 명령을 한 행의 명령으로 간단히 만들수 있는가?

```
if(B1 == false)
  if(B2 == false)
    System.out.println("거짓");
  else
    System.out.println("참");
else
  System.out.println("참");
```

2.2.14. 옹근수값 0, 1을 론리형자료 false, true로 변환하는 메쏘드를 작성하시오.

2.2.15. 론리형의 형식파라메터를 접수하고 만일 파라메터값이 거짓이면 《안녕하십니까!》를 인쇄하고 파라메터값이 참이면 《안녕히 계십시오.》를 인쇄하는 메쏘드를 작성하시오.

2.2.16. 우의 문제를 수정하여 만일 파라메터값이 거짓이면 메쏘드는 《안녕하십니까!》를 되돌려주고 그렇지 않으면 《안녕히 계십시오.》를 되돌려주게 하시오. 두 메쏘드의 차이와 각자가 리용한 상태를 비교하시오.

2.2.17. 사용자가 건반으로 입력한 2개의 옹근수를 접수하고 그의 총합을 출력하기 위하여 어떤 학생이 아래의 코드를 작성하였다. 틀린것을 지적하고 수정하시오.

```
import java.io.*;
public class Ch3ErrorCorrent1
```

```
{
  public static void main(String args[ ])
  {
    try
    {
      BufferedReader br = new BufferedReader(new InputStreamReader(System. in));
      System.out.println("매 수가 한개행씩 차지하도록 정의옹근수를 입력하시오.");
      System.out.println(br.readLine( ) + br.readLine( ));
    }
    catch(IOException ioe)
    {
      ioe.printStackTrace( );
    }
  }
}
```

2.2.18. 아래의 문장에서 틀린것이 있는가를 판정하고 수정하시오.

(1)
```
if (x > 10)
  x = 0;
else;
  x = 1;
```

(2)
```
if(isTrue = true);
  system.out.println("It's true!");
```

(3)
```
int m = 9;
switch(m / 3.0)
{
  case0: System.out.println("it is zero");
  case1: System.out.println("it is one");
  default: System.out.println("it is not zero or one");
}
```

2.2.19. 《Java메쏘드에서 형식파라메터와 실제파라메터를 결합하여 파라메터가 기본자료형의 변수일 때는 값주기이고 파라메터가 객체일 때는 인용값주기이다.》라는 말이 옳은가?

2.2.20. 어떤 해의 첫번째 날이 월요일이다. 사용자가 입력한 1~365의 한개 수자 N을 접수하여 이 날에 해당한 날자와 요일을 출력하시오. 실례로 2를 입력하면 《1월 2일, 화요일》을 출력한다.

2.2.21. 사용자가 입력한 3개의 옹근수를 립방체의 세변으로 하는 립방체의 겉면적과 체적을 계산하고 출력하는 프로그람을 작성하시오.

2.2.22. 2개의 본문마당으로 2개의 옹근수를 접수하여 첫번째 옹근수가 두번째 옹근수로 나누어지는가를 검사하고 결과를 출구하는 도형대면의 Java Applet를 작성하시오.

2.2.23. 대면이 나누어지는 수를 입력하는 본문마당, 나누는 수를 입력하는 본문마당, 결과를 출력하는 표식자(label)를 포함하는 류동소수점수나누기를 Java Applet로 작성하시오. 2개의 본문마당에 류동소수점수를 입력한 다음 나누는 수를 입력한 본문마당에서 넣기건(Enter건)을 누르면 연산결과가 얻어진다. 가령 나누는 수가 0일 때 어떤 결과를 얻을수 있는가?

2.2.24. 론리형파라메터를 접수하여 대응하는 문자렬 《TRUE》와 《FALSE》을 되돌려주는 메쏘드를 작성하시오.

2.2.25. 2개의 본문마당과 1개의 표식자를 포함하는 Applet프로그람을 작성하시오.

거기서 사용자가 입력한 2개의 옹근수를 접수하여 두 수의 최소공통배수와 최대공통약수를 표식자에 출력하는 프로그람을 작성하시오.

2.2.26. 다음 문장에서 틀린것을 지적하시오.

(1)
```
for(int i = 0, i < 10, i++)
  System.out.print(i + '/t');
```
(2)
```
for(int i = 0; i == 10; i++;)
  System.out.print(i + "/t");
```
(3)
```
for(int i = 0; i = 10; i++)
  System.out.print(i + "/t");
```
(4)
```
for(int i = 0; i < 10; i--)
  System.out.print(i+"/t");
```
(5)
```
for(int i = 0; I < 10; i--);
  System.out.print(i + "/t");
```
(6)
```
int i = 0;
while(i < 10)
  System.out.print(i + "/t");
```
(7)
```
int i = 0;
do
{
  System.out.print(i++ + "/t");
} while(i < 10)
```
(8)
```
int i = 0;
do
```

```
    {
      System.out.print(i++ + "/t");
    };
```

(9)
```
int i = 0;
while(i < 10)
  System.out.print(i + "/t");
i++;
```

2.2.27. 아래의 도형을 인쇄하는 프로그람을 작성하시오.

1)
```
*
* *
* * *
* * * *
* * * * *
```

2)
```
* * * * *
* * * *
* * *
* *
*
```

3)
```
*
  *
    *
      *
        *
```

2.2.28. 수자 1～9의 곱하기표를 인쇄하는 프로그람을 작성하시오.

2.2.29. 아래의 메소드를 읽고 그 기능을 말하시오. 이 메소드의 되돌이값이 파라메터값과 관계가 있는가?

```
int myMethod(int x)
{
  while(x > 0)
  {
    if(x % 2 == 1)
      x = (x - 1) / 2;
    else
      x = x / 2;
    System.out.println(x + "\t");
  }
  return x;
}
```

2.2.30. 어떤 수가 세자리수로 되여있고 그의 매 자리수의 세제곱의 합이 자기 자체와 같은 모든 수를 찾고 출력하는 Application프로그람을 작성하시오. 각각 while순환과 for순환을 리용하여 실현하시오.

2.2.31. 아래의 식을 리용하여 $e^x$을 계산하시오.

$e^x = x + x / 1! + x^2 / 2! + x^3 / 3! + x^4 / 4! + \cdots + x^n / n!$

2.2.32. -1이 입력의 끝을 표시하며 사용자가 입력한 옹근수들을 접수하여 그의 최대값과 최소값, 평균값을 구하는 프로그람을 작성하시오.

# 제3장. 추상과 내장, 클라스

## 3.1. 련습

### 3.1.1. 추상, 수속추상, 자료추상이란 무엇이며 객체지향쏘프트웨어개발에서 추상을 어떻게 실현하는가?

추상은 객체연구에서 기본목적과 관계없는 부차적인 혹은 얼마동안 고려하지 않아도 되는 부분을 제거하고 오직 연구사업과 관계가 있는 실질적인 내용을 추려 고찰하는 과학연구방법이다. 수속추상은 전체 체계의 기능을 몇개 부분으로 나누고 기능을 완성하는 과정과 단계이다. 자료추상은 체계에서 처리하려는 자료와 이 자료의 조작을 함께 결합하여 그 기능, 성질, 작용 등의 인자에 근거하여 서로 다른 추상자료형으로 추상화된다. 매 추상자료형은 자료를 포함할뿐만아니라 이 자료에 해당한 권한이 부여된 조작을 포함한다. 자료추상은 수속추상에 비하여 보다 엄격하고 합리적인 추상방법이다.

객체지향쏘프트웨어개발에서는 자료추상방법을 리용하여 프로그람의 클라스, 객체, 메쏘드를 만든다.

### 3.1.2. 내장이란 무엇이며 객체지향프로그람설계에서 내장을 어떻게 실현하는가?

내장은 추상자료형을 리용하여 자료와 자료에 기초한 조작을 함께 매몰시켜 자료가 추상자료형의 내부에서 보호되게 하며 체계의 다른 부분은 오직 자료밖에 있는 권한이 부여된 패키지의 조작을 통해서만 이 추상자료형와 호상 작용할수 있다.

객체지향프로그람설계에서 추상자료형은 《클라스》라는 객체지향도구가 리해하고 조종할수 있는 구조를 리용하여 표현한것이다. 매개 클라스에는 관계되는 자료와 조작을 모두 내장하였다.

### 3.1.3. 추상과 내장을 리용하면 어떤 좋은점이 있는가?

추상을 리용하면 기본문제와 관계가 없는 부차적인것을 잠시 삭제하여 개발로 하여금 비교적 기본적이고 중요한 부분에 집중할수 있게 한다. 그러므로 개발의 중점을 명확히 하고 문제의 본질을 정확히 리해할수 있다. 내장을 리용하면 자료의 안전성, 체계의 엄밀성, 개발패키지의 재리용가능성을 높일수 있으므로 개발과정의 복잡성을 없애고 개발효과와 질을 높일수 있다.

### 3.1.4. Java프로그람에서 리용하는 클라스에는 어떤것이 있으며 체계정의클라스와 사용자정의클라스란 무엇인가?

Java프로그람에서 리용하는 클라스는 체계정의클라스와 사용자정의클라스로 나눈다.

체계정의클라스는 Java언어가 자체로 가지고있는 이미 정의된 클라스이다. 즉 Java클라스서고의 클라스이다. 사용자정의클라스는 Java언어를 리용하는 개발자들이 해결하려는 문제에 근거하여 자체로 설계하고 정의하는 클라스이다.

**3.1.5. 클라스서고란 무엇이며 왜 Java프로그람작성에서 클라스서고를 잘 알아야 하는가? 자주 리용하는 5개의 클라스서고패키지를 말하시오.**

Java의 클라스서고는 여러 개발자들 또는 쏘프트웨어개발집단에 의하여 작성된 Java프로그람패키지이다. 매개 패키지에는 자체의 특수한 기능이 포함되여 있다.

Java클라스서고를 잘 알기만 하면 서고에 있는 기능을 충분히 리용할수 있으며 훌륭한 코드를 작성할수 있다.

**3.1.6. 이미 존재하는 클라스를 리용하는데 어떤 중요한 방법이 있는가? 이미 존재하는 클라스를 프로그람에 어떻게 인입하는가?**

이미 존재하는 클라스를 리용하는 첫번째 방법은 그것을 계승하는것이며 사용자프로그람에서 그의 하위클라스를 만드는것이다. 두번째 방법은 그의 객체를 만드는것이다. 세번째 방법은 이미 존재하는 클라스의 정적마당과 정적메쏘드를 직접 리용하는것이다.

이미 존재하는 클라스를 리용하기전에 그 클라스가 현재 프로그람에서 볼수 있는것인가를 확인하여야 한다. 만일 이 클라스가 현재 프로그람자체 또는 현재 프로그람에 있는 패키지안의 어떤 프로그람에 의하여 정의된것이라면 이 클라스는 현재 프로그람에서 볼수 있는것이다. 만일 이 클라스가 다른 패키지에 의하여 정의된것이라면 현재 프로그람은 이 클라스를 리용하기전에 먼저 import명령을 리용하여 현재 프로그람에서 사용해야 할 클라스를 정의한 그 패키지를 내리적재하여야 한다.

**3.1.7. 학생을 표시하는 클라스 Student를 정의하는 Java프로그람을 작성하시오. 마당은 《학생번호》, 《학급번호》, 《이름》, 《성별》, 《나이》이다. 메쏘드는《학생번호얻기》, 《학급번호얻기》, 《성별얻기》, 《이름얻기》, 《나이얻기》, 《나이고치기》이다.**

```
원천프로그람 Student.java
class Student
{
  private String studentId;
  private int classId;
  private String studentName;
  private char studentGender;
  private int studentAge;

  String getStudentId()
```

```
    {
      return studentId;
    }
    int getClassId( )
    {
      return classId;
    }
    char getStudentGender( )
    {
      return studentGender;
    }
    String getStudentName( )
    {
      return studentName;
    }

    int getStudentAge( )
    {
      return studentAge;
    }
    boolean setStudentAge(int newAge)
    {
      if(newAge > 0 && newAge < 130)
      {
        studentAge = newAge;
        return true;
      }
      else
      {
        System.out.println("학생나이가 아니다.");
        return false;
      }
    }
}
```

3.1.8. 문제 3.1.7에 기초하여 Student클라스의 객체를 만드는 Java Application프로그람을 작성하시오.

```
원천프로그람 ch3_e3_8.java
public class ch3_e3_8
{
  public static void main(String args[ ])
  {
    Student demoStudent = new Student( );
  }
}

class Student
{
  private String studentId;
  private int classId;
  private String studentName;
  private char studentGender;
  private int studentAge;
  String getStudentId( )
  {
    return studentId;
  }
  int getClassId( )
  {
    return classId;
  }
  char getStudentGender( )
  {
    return studentGender;
  }
  String GetStudentName( )
  {
  return studentName;
  }
  int getStudentAge( )
  {
```

```
    return studentAge ;
    }
    boolean setStudentAge(int newAge)
    {
      if(newAge > 0 && newAge < 130)
      {
        studentAge = newAge;
        return true;
      }
      else
      {
        System.out.println("학생나이가 아니다.");
        return false;
      }
    }
  }
```

**3.1.9.** Student클라스에 대하여 구성자를 정의하고 모든 마당을 초기화하시오. 메쏘드 public String toString( )을 추가하여 Student클라스객체의 모든 마당의 정보를 하나의 문자렬로 만드시오. 새로 추가된 기능을 포함하는 Java Application프로그람을 작성하시오.

```
원천프로그람 ch3_e3_9.java
public class ch3_e3_9
{
  public static void main(String args[ ])
  {
    Student demoStudent = new Student("008", 892, 'm', "장산", 19);
    System.out.println(demoStudent.toString( ));
  }
}

class Student
{
  private String studentId;
  private int classId;
  private String studentName;
  private char studentGender;
```

```
private int studentAge;
String getStudentId( )
{
  return studentId;
}
int getClassId( )
{
  return classId;
}
char getStudentGender( )
{
  return studentGender;
}
String getStudentName( )
{
  return studentName;
}
int getStudentAge( )
{
  return studentAge;
}
boolean  setStudentAge (int  newAge)
{
  if ( newAge > 0 && newAge < 130)
  {
    studentAge = newAge;
    return true;
  }
  else
  {
    System.out.println("학생나이가 아니다.");
    return false;
  }
}
Student(String id, int clsId, char gender, String name, int age)
{
  studentId = new String(id);
```

```
      classId = clsId;
      studentName = new String(name);
      studentAge = age;
      studentGender = gender;
    }
    public String toString( )
    {
      return "학생정보: \n" + "학생번호: " + this.getStudentId( )
        + "\t학급번호: " + this.getClassId( ) + "\t이름: " + this.getStudentName( )
        + "\t성별: "+((this.getStudentGender( )) == 'm'? "남": "녀")
        + "\t나이: " + this.getStudentAge( );
    }
  }
```

**3.1.10. 구성자의 기능과 특징을 간단히 서술하시오. 아래의 프로그람은 어떤 학생이 Student클라스에 대하여 만든 구성자이다. 어느곳이 틀리는가?**

```
void Student(int sno, String sname)
{
   studentNo = sno;
   studentName = sname;
   return sno;
}
```

구성자는 일종의 특수한 메쏘드이다. 매개 클라스는 하나 혹은 여러개의 구성자를 가질수 있다. 프로그람에서 어떤 클라스의 객체를 만들 때 체계는 그 클라스를 실행하는 구성자를 자동적으로 리용할수 있다. 구성자에서는 새로 만들어진 객체에 대한 초기화조작을 정의한다.

구성자의 메쏘드이름과 그것이 있는 클라스이름은 같다. 구성자는 되돌이값이 없다. 일반적으로 프로그람작성자에 의하여 직접 호출될수 없다.

프로그람은 두개의 틀린곳이 있다. 구성자는 되돌이값을 가지지 않으며 또 되돌이조작을 요구할수 없다. 수정한 프로그람은 다음과 같다.

```
Student(int sno, String sname)
{
   studentNo = sno;
   studentName = sname;
}
```

3.1.11. 사용자가 입력한 카드번호, 암호, 금액, 받는 사람번호를 접수하여 PhoneCard 클라스의 객체를 만들고 이 전화카드와 관계되는 정보를 출력하는 도형대면의 Applet를 작성하시오.

```
원천프로그람 ch3_e3_11.java
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch3_e3_11 extends Applet implements ActionListener
{
    Label numberPrompt;
    Label passwordPrompt;
    Label balancePrompt;
    Label connectNumberPrompt;
    TextField numberTfd;
    TextField passwordTfd;
    TextField balanceTfd;
    TextField connectNumberTfd;
    TextArea infoSummary;
    Button createPhoneCardBtn;
    PhoneCard myCard;

    public void init( )
    {
        numberPrompt = new Label("카드번호");
        passwordPrompt = new Label("암호번호");
        balancePrompt = new Label("나머지 금액");
        connectNumberPrompt = new Label("받는 사람의 번호");
        numberTfd = new TextField(10);
        passwordTfd = new TextField(5);
        passwordTfd.setEchoChar('*');
        balanceTfd = new TextField(5);
        connectNumberTfd = new TextField(5);
        infoSummary = new TextArea(8,30);
        createPhoneCardBtn = new Button("만들기");
        add(numberPrompt);
        add(numberTfd);
```

```
        add(passwordPrompt);
        add(passwordTfd);
        add(balancePrompt);
        add(balanceTfd);
        add(connectNumberPrompt);
        add(connectNumberTfd);
        add(createPhoneCardBtn);
        add(infoSummary);
        createPhoneCardBtn.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            myCard = new PhoneCard(
            Long.parseLong(numberTfd.getText( )),
            Integer.parseInt(passwordTfd.getText( )),
            Double.parseDouble(balanceTfd.getText( )),
            connectNumberTfd.getText( ));
            infoSummary.setText(myCard.toString( ));
        }
        catch(NumberFormatException nfe)
        {
            infoSummary.setText("입력자료형식이 틀리면 다시 입력합니다.");
            numberTfd.setText(" ");
            passwordTfd.setText(" ");
            balanceTfd.setText(" ");
            connectNumberTfd.setText(" ");
        }
    }
}

class PhoneCard
{
    long cardNumber;
    private int password;
    double balance;
```

```
String connectNumber;
boolean connected;
PhoneCard(long cn, int pw, double b, String s)
{
   cardNumber = cn;
   password = pw;
   if(b > 0)
   balance = b;
   else
   System.exit(1);
   connectNumber = s;
   connected = false;
}
boolean performConnection(long cn, int pw)
{
   if(cn == cardNumber && pw == password)
   {
      connected = true;
      return true;
   }
   else
   {
      connected = false;
      return false;
   }
}
double getBalance( )
{
   if(connected)
      return balance;
   else
      return -1;
}
void performDial( )
{
   if(connected)
      balance -= 0.5;
```

```
    }
    public String toString( )
    {
       String s = "전화카드 받는 사람번호: " + connectNumber + "\n전화카드번호: "
         + cardNumber + "\n전화카드암호: " + password + "\n나머지 금액: " + balance;
       if(connected)
         return(s + "\n전화는 통화중입니다.");
       else
         return(s + "\n전화가 아직 되지 않습니다.");
    }
}
```

### 3.1.12. 장식부란 무엇이며 어떤 작용을 하는가? 왜 추상클라스를 정의해야 하는가? 추상클라스의 정의방법과 실례를 드시오.

장식부는 수식되지 않은 클라스, 객체, 메쏘드 혹은 마당의 속성을 설명하는데 리용되는 체계가 정의하는 기호이다.

추상클라스는 구체적인 객체가 없는 개념클라스이다. 추상클라스를 구체적인 객체를 가진 클라스의 공통속성모임으로 리용하면 보다 편리하고 유리하게 클라스구조를 만들수 있으며 개발과 보호의 효률을 높일수 있다. 또한 추상클라스를 리용하면 사람들의 일상사고습관에 부합되므로 프로그람의 읽기가능성을 높일수 있다. 그러므로 추상클라스는 객체지향기술을 인류의 사고방식에 모방한 구체적인 실례이다.

abstract장식부를 리용하여 추상클라스를 정의한다.

현실에서 추상클라스의 실례는 사람, 자동차, 동물 등이다.

### 3.1.13. 최종클라스란 무엇이며 그것을 어떻게 정의하는가? 최종클라스의 실례를 드시오.

최종클라스는 하위클라스를 가지지 않는 클라스이다. 최종클라스는 클라스의 계승관계나무구조에서 나무잎이라고 볼수 있다. 일반적으로 최종클라스를 리용하여 고정되고 쉽게 고칠수 없는 표준조작을 정의한다.

Java프로그람은 final장식부를 리용하여 최종클라스를 정의한다.

실례로 체계클라스 System에는 많은 표준조작이 정의되여 있으며 사용자프로그람에서 그에 대하여 파생과 내리적재를 허용하지 않는것을 들수 있다. 그리고 서로 다른 기능을 정의하여 실행할 때 예견할수 없는 문제가 생기지 않도록 한다.

그러므로 System클라스는 Java클라스서고에서 public final class System으로 정의된다.

### 3.1.14. 정적마당을 어떻게 정의하며 어떤 특징을 가지는가? 정적마당의 자료에 어떻게 접근하고 수정하는가?

Java에서 static장식부를 리용하여 정적마당을 정의한다. 정적마당은 클라스에 속하지만 클라스의 어떤 객체의 마당이 아니다. 정적마당은 클라스의 기억구역에 보관되지만 어떤 객체가 전문적으로 속한 기억기구역에는 보관되지 않는다. 정적마당에 접근하고 수정하려면 그가 속한 클라스이름을 리용해야 하지만 어떤 하나의 객체이름을 앞붙이로 가져서는 안된다.

### 3.1.15. 정적초기화란 무엇이며 어떤 특징이 있는가? 정적초기화와 구성자의 다른 점은 무엇인가?

정적초기화는 클라스정의에서 예약어 static에 의하여 설명되는 대괄호로 묶은 명령묶음이다. 체계가 클라스를 적재할 때 이 클라스안의 정적초기화를 자동적으로 리용하여 클라스에 대한 초기화를 할수 있다. 정적초기화와 구성자는 류사하며 다 같이 초기화를 하는데 리용된다.

그러나 정적초기화는 클라스를 초기화하는데 리용되며 구성자는 클라스의 어떤 구체적인 객체를 초기화하는데 리용된다. 정적초기화와 구성자들을 체계가 자동리용하는 시기가 서로 다르다. 또한 정적초기화는 단지 하나의 명령묶음이며 구성자와 같이 메쏘드이름과 파라메터렬을 가진 하나의 완전한 메쏘드는 아니다.

### 3.1.16. 최종마당과 휘발성마당은 어떤 마당이며 정의방법을 말해보시오.

최종마당은 프로그람실행과정에 수값이 변하지 않는 마당이며 휘발성마당은 수값이 현재 실행되는 프로그람밖의 다른 프로그람 혹은 체계에 의하여 변할수 있는 마당이다.

Java프로그람에서 최종마당은 final로, 휘발성마당은 volatile로 정의한다.

### 3.1.17. 메쏘드를 어떻게 정의하며 객체지향프로그람설계에서 메쏘드는 어떤 작용을 하는가?

메쏘드정의와 기타성분정의의 가장 기본적인 차이는 메쏘드정의가 반드시 대괄호를 리용한다는데 있다. 메쏘드정의의 일반적형식은 다음과 같다.

```
장식부1 장식부2 …… 되돌이값형 메쏘드이름(형식파라메터렬) throw [례외목록]
{
   메쏘드본체의 명령문들;
}
```

객체지향프로그람설계에서는 메쏘드를 리용하여 클라스안의 내장된 자료들에 대한 여러가지 조작을 정의한다.

### 3.1.18. 추상메쏘드란 무엇이며 어떤 특징을 가지는가? 추상메쏘드를 어떻게 정의하고 리용하는가?

추상메쏘드는 abstract장식부로 정의하는 메쏘드이다. 추상메쏘드는 오직 메쏘드머리부만을 가지며 구체적인 메쏘드본체는 가지지 않는다.

추상메쏘드를 리용하면 같은 상위클라스의 모든 하위클라스에서 통일적인 외부접속을 실현할수 있다. 동시에 메쏘드에서 구체적인 기능을 실현하는 코드세부는 밀봉되여있다. 추상메쏘드는 반드시 추상클라스에서 정의하며 그 추상클라스의 하위클라스는 상위클라스의 추상메쏘드를 내리적재하고 메쏘드본체에 그의 구체적인 내용을 서술한다.

### 3.1.19. 정적메쏘드를 어떻게 정의하며 어떤 특징을 가지는가? 정적메쏘드가 처리하는 마당은 무엇을 요구하는가?

정적메쏘드를 정의하려면 static장식부를 리용해야 한다. 정적메쏘드는 전체 클라스에 속하는 메쏘드이기때문에 동일한 전체클라스에 속하는 정적마당만을 처리할수 있다.

### 3.1.20. 최종메쏘드, 고유메쏘드, 동기메쏘드의 정의와 어느 경우에 리용하는가를 간단히 서술하시오.

Java프로그람에서 final장식부를 리용하여 최종메쏘드를 정의한다. 최종메쏘드는 조작이 고정되고 수정과 내리적재를 할수 없는 메쏘드이다.

Native장식부가 수식하는 메쏘드를 고유메쏘드라고 한다. 고유메쏘드는 Java프로그람에서 오직 메쏘드머리부의 정의만을 가지며 그의 구체적인 메쏘드본체는 항상 다른 언어 즉 C, C++, FORTRAN 등에 의하여 다른 프로그람에서 정의된다. Java프로그람에서 고유메쏘드를 정의하는것은 이미 존재하는 프로그람기능모듈을 충분히 리용하자는것과 반복작업을 피하자는데 있으며 프로그람의 실행속도를 높이자는데 있다.

Synchronized장식부가 수식하는 메쏘드를 동기메쏘드라고 부른다. 동기메쏘드는 주로 다중처리조작에서 자원을 공유하고 처리를 협조하는데 리용한다.

### 3.1.21. 접근조종부란 무엇이며 어떤 접근조종부들이 있는가? 어떤것이 클라스와 마당 및 메쏘드를 장식하는데 리용되는가? 이 접근조종부들의 작용을 말하시오.

접근조종부는 클라스와 마당을 제한하거나 혹은 메쏘드가 프로그람의 다른 부분들에 의하여 접근되고 리용될수 있는가 없는가를 결정하는 장식부이다. 접근조종부에는 공개접근조종부(public), 기정접근조종부, 비공개접근조종부(private), 보호접근조종부(protected) 등이 있다.

공개접근조종부와 기정접근조종부는 클라스를 장식하는데 리용할수 있다. 모든 접근조종부들은 마당과 메쏘드를 장식하는데 리용할수 있다.

공개접근조종부가 장식하는 성분은 어느것이나 그에 접근할수 있으며 제한을 주지 않는다.

기정접근조종부는 오직 동일한 패키지의 성분일 때만 그가 장식하는 성분에 접근할수 있다.

비공개접근조종부는 오직 동일한 클라스의 성분일 때만 그가 장식하는 성분에 접근할수 있다.

보호접근조종부는 기정접근조종부가 허용하는 성분외에 다른 패키지에서 현재클라스의 하위클라스도 접근조종권한을 가진다.

마당과 메쏘드의 접근조종권한을 론할 때 자기의 접근조종부외에 동시에 그가 있는 클라스의 접근조종부의 제한작용을 고려하여야 한다.

### 3.1.22. 장식부를 혼합하여 리용할수 있는가? 혼합하여 리용할 때 어떤 문제에 주의해야 하는가?

기능을 표현하는 서로 다른 장식부는 혼합하여 리용할수 있다. 그러나 일부 장식부를 혼합하여 리용하면 서로 모순되는 성분으로 하여금 의미가 없어지기때문에 이러한 경우를 피하여야 한다. 혼합하여 리용할수 없는 장식부는 다음과 같다.

1) abstract는 final과 함께 클라스를 장식할수 없다.

2) abstract는 private, static, final, native와 함께 메쏘드를 장식할수 없다.

3) abstract클라스에는 private의 성원(속성과 메쏘드를 포함)이 있을수 없다.

4) abstract메쏘드는 반드시 abstract클라스에 있어야 한다.

5) static메쏘드에서 static가 아닌 마당을 처리할수 없다.

## 3.2. 보충문제

3.2.1. 만일 구성자를 리용하지 않으면 어떤 문제가 제기될수 있는가를 실례를 들어 설명하시오.

3.2.2. 아래의 개념들이 서로 다른 점을 말하시오.

1) 변수정의, 객체창조

2) 객체의 실례를 창조, 객체의 인용을 정의

3.2.3. 임의의 형식의 파라메터도 가지지 않는 구성자를 기정구성자라고 부른다. 오직 하나의 동일한 클라스의 객체를 형식파라메터로 하고 형식파라메터객체의 매개 마당의 값을 새로 만든 객체에게 복사하여 주는 구성자를 복제구성자라고 부른다.

아래의 클라스에 대하여 기정구성자와 복제구성자를 정의하시오.

```
class MyClass
{
   double x;
   int y;
   String str;
}
```

3.2.4. 어떤 학생이 자체로 정의한 클라스 MyClass에 대하여 아래와 같은 구성자를 만들었다. 틀린곳을 찾아내시오.

```
public void MyClass( )
{
   System.out.println("Create a new instance of MyClass!");
}
```

3.2.5. 구성자를 클라스의 다른 메쏘드에서 리용할수 있는가? 다른 클라스의 메쏘드에서 리용할수 있는가?

3.2.6. 인쇄감시기를 실현하는 코드를 작성하시오. 거기에 2개의 론리형마당 《잉크카트리지》, 《마지막으로 수리한후 100만장의 종이를 인쇄》와 메쏘드《수리를 요구한다.》를 포함한다. 그 메쏘드는 2개의 론리형마당에서 하나가 참으로 될 때 되돌이값을 가진다.

3.2.7. 온도클라스를 실현하는 코드를 작성하시오. 거기에 4개의 메쏘드가 포함된다. getFahrenheit( )와 getCelsius( )는 각각 화씨온도값과 섭씨온도값을 되돌려주며 setFahrenheit( )와 setCelsius( )는 각각 화씨온도값과 섭씨온도값을 설정한다.

3.2.8. 주소클라스를 실현하는 코드를 작성하시오. 매개 객체에는 《이름》, 《주소》, 《전화》인 3개마당이 포함되고 매개 마당에 해당한 get, set메쏘드를 정의한다.

3.2.9. 은행장부클라스를 실현하는 코드를 작성하시오. 거기에 장부번호, 이름, 장부사용날자, 저금과 남은돈 등 몇개 마당 및 해당한 조작이 포함된다.

# 제4장. 계승과 다형성

## 4.1. 련습

**4.1.1. 계승, 상위클라스, 하위클라스란 무엇이며 계승의 특징은 객체지향프로그람작성에서 어떤 우점을 가지는가? 단일계승과 다중계승이란 무엇인가?**

계승은 객체지향프로그람설계에서 2개의 클라스들사이에 존재하는 관계이다. 하나의 클라스가 다른 클라스의 모든 공개적인 자료와 조작의 정의를 자기의 부분 혹은 전체성분으로 할 때 이 2개 클라스들사이에는 계승관계가 있다고 말한다.

계승되는 클라스를 상위클라스 혹은 초클라스라고 한다. 상위클라스의 모든 자료와 조작을 계승한 클라스를 하위클라스라고 한다. 객체지향프로그람설계에서 계승은 프로그람구조를 보다 간단하고 명백히 하며 코드작성과 보호의 작업량을 낮춘다.

만일 한개 클라스가 하나의 상위클라스만을 가질수 있다면 이 계승관계를 단일계승이라고 부르며 한개 클라스가 여러개의 상위클라스를 가질수 있다면 이 계승관계를 다중계승이라고 부른다.

**4.1.2. 계승관계를 어떻게 정의하는가? 문제 3.1.7에서 정의한 학생클라스는 《소학생》, 《중학생》, 《대학생》, 《연구생》인 4개의 클라스로 파생된다. 그중 《대학생》클라스는 다시 《1학년학생》, 《2학년학생》, 《3학년학생》, 《4학년학생》인 4개의 하위클라스로, 《연구생》클라스는 《학사반생》, 《박사반생》인 2개의 하위클라스로 파생되여 나온다.**

클라스를 정의할 때 extends예약어로 새롭게 정의하는 클라스의 상위클라스를 지정함으로써 2개의 클라스들사이의 계승관계를 정의한다.

```
상위클라스:
Student
하위클라스:
public class ElementaryStudent extends Student
{
}
public class HighSchoolStudent extends Student
{
}
public class CollegeStudent extends Student
{
}
```

```
public class GraduateStudent extends Student
{
}
public class Freashman extends CollegeStudent
{
}
public class Sophomore extends CollegeStudent
{
}
public class Junior extends CollegeStudent
{
}
public class Senior extends CollegeStudent
{
}
public class MasterStudent extends GraduateStudent
{
}
public class PhDStudent extends GraduateStudent
{
}
```

**4.1.3. 아래의 프로그람을 보고 그의 상위클라스와 하위클라스 그리고 상위클라스와 하위클라스의 매개 마당과 메쏘드를 지적하시오.**

```
Class SuperClass
{
   int data;
   void setData(int newData)
   {
      data = newData;
   }
   int getData( )
   {
      return data;
   }
}
class SubClass extends SuperClass
```

```
{
   int subData;
   void setSubData(int newData)
   {
      subData = newData;
   }
   int getData( )
   {
      return subData;
   }
}
```

상위클라스 SuperClass

상위클라스의 마당 data

상위클라스의 메쏘드 setData( ), getData( )

하위클라스 SubClass

하위클라스의 마당 data, subData

하위클라스의 메쏘드  setSubData( ), getData( )

**4.1.4. 하위클라스의 마당과 메쏘드의 개수는 상위클라스의 마당과 메쏘드의 개수와 같거나 크다는 말이 정확한가? 그 리유를 말하시오.**

이 말은 틀린다. 그 리유는 하위클라스가 상위클라스의 모든 마당과 메쏘드를 반드시 계승하는것이 아니며 오직 모든 공개적인 마당과 메쏘드만을 계승할수 있기때문이다.

만일 하위클라스가 계승하고있는것 외에 자체로 정의한 마당과 메쏘드의 개수가 상위클라스에서 비공개성원의 개수보다 작다면 하위클라스의 마당과 메쏘드의 개수는 상위클라스의 마당과 메쏘드의 개수보다 작다.

**4.1.5. 자료의 밑봉이란 무엇이며 만일 문제 4.1.3의 SubClass에서 변수 data를 다시 정의하면 SubClass클라스에 어떤 마당들이 포함되는가?**

만일 하위클라스에서 상위클라스에 이미 존재하는 마당과 이름이 같은 형의 마당을 새로 정의하면 하위클라스가 상위클라스로부터 계승한 새로 정의한 마당과 이름이 같은 마당은 밑봉된 마당으로 변화된다.

만일 문제 4.1.3의 SubClass에서 상위클라스에 이미 있는 마당 data를 정의하였다면 그것은 3개의 마당 subData, 상위클라스가 정의한 data, 하위클라스자체가 정의한 data를 포함한다.

**4.1.6. 메쏘드의 다중정의란 무엇이며 메쏘드의 다중정의와 마당의 밑봉은 어떻게 차이나는가? 메쏘드의 내리적재와 어떤 차이가 있는가?**

하위클라스에서 상위클라스에 이미 있는 메쏘드를 새로 정의하는것을 메쏘드의 다중정의라고 부른다.

메쏘드의 다중정의와 마당의 밑봉은 서로 다르다.

하위클라스에서 상위클라스에 이미 있는 마당을 새로 정의하면 하위클라스가 상위클라스로부터 계승한 이름이 같은 마당을 완전히 대신할수 없다. 이 마당은 여전히 하위클라스의 내부기억공간을 차지하며 어떤 조건에서는 리용될수 없다.

그러나 하위클라스에서 상위클라스의 메쏘드를 새로 정의할 때 상위클라스로부터 계승한 메쏘드는 새로운 메쏘드에 의하여 완전히 대신할수 있으며 하위클라스의 내부기억공간을 일시적으로 차지할수 없다.

메쏘드의 다중정의와 메쏘드의 내리적재는 서로 다르다. 내리적재는 상위클라스와 이름이 같은 메쏘드에 대한 하위클라스에서의 새로운 정의가 아니라 한개 클라스에서 이름이 같은 서로 다른 메쏘드를 정의한것이다.

**4.1.7. 문제 4.1.3의 SubClass에서 메쏘드의 다중정의를 어떻게 실현하는가?**

SubClass에서 상위클라스 SuperClass에 이미 있는 메쏘드인 getData( )를 새로 정의하는것으로써 메쏘드의 다중정의를 실현한다.

**4.1.8. this, super의 의미와 작용을 해설하시오.**

예약어 this는 현재클라스의 현재객체를 지적한다. 예약어 super는 현재클라스의 직접적인 상위클라스의 객체를 지적한다.

**4.1.9. 다형성이란 무엇이며 객체지향프로그람설계에서 왜 다형성의 특징을 도입하여야 하는가? 다형성을 리용하면 어떤 우점이 있는가?**

한개 클라스에 이름이 같은 여러개의 메쏘드가 존재하는것을 다형성이라고 부른다.

추상기능과 목적이 서로 같고 서로 다른 조건에서 서로 다른것을 구체적으로 실현하는 메쏘드를 이름이 같은 메쏘드라고 하면 객체지향프로그람설계에서 추상, 승급개념, 내장, 밑봉세부의 특징에 부합되므로 프로그람의 간결성과 읽기가능성을 높이고 클라스와 프로그람묘듈의 직접적인 호상융합과 의존을 낮춘다.

**4.1.10. Java프로그람은 어떻게 다형성을 실현하는가? 어떤 방식이 있는가?**

Java프로그람에서는 다중정의와 내리적재의 2가지 방식을 리용하여 다형성을 실현한다. 다중정의는 하위클라스에서 상위클라스에 이미 있는 메쏘드를 새로 정의하는 방식이며 내리적재는 한개 클라스내부에서 이미 있는 메쏘드와 이름이 같지만 파라메터렬이 서로 다른것을 정의하는 방식을 말한다.

**4.1.11. 아래의 요구에 근거하여 복소수클라스 ComplexNumber를 실현하는 프로그람을 작성하시오.**

1) 복소수클라스 ComplexNumber의 속성

m_dRealPart: 복소수의 실수부를 표시한다.

m_dImaginPart: 복소수의 허수부를 표시한다.

2) 복소수클라스 ComplexNumber의 메쏘드

ComplexNumber( ): 구성자인데 실수부, 허수부를 모두 0으로 놓는다.

ComplexNumber(double r, double i): 구성자인데 복소수객체를 만드는 동시에 복소수의 실수부 허수부의 초기화를 한다. r는 실수부의 초기값, i는 허수부의 초기값이다.

getRealPart( ): 복소수객체의 실수부를 얻는다.

getImaginaryPart( ): 복소수객체의 허수부를 얻는다.

setRealPart(double d): 복소수객체의 실수부를 주어진 형식파라메터의 수자로 놓는다.

setImaginaryPart(double d): 복소수객체의 허수부를 주어진 형식파라메터의 수자로 놓는다.

complexAdd(ComplexNumber c): 현재 복소수객체와 형식파라메터복소수객체를 더한 결과 역시 복소수값이며 그 메쏘드의 리용자에게 되돌려준다.

complexAdd(double c): 현재 복소수객체와 형식파라메터실수객체를 더한 결과 역시 복소수값이고 그 메쏘드의 리용자에게 되돌려준다.

complexMinus(ComolexNumber c): 현재 복소수객체와 형식파라메터복소수객체를 호상 던 결과 역시 복소수값이고 그 메쏘드의 리용자에게 되돌려준다.

complexMinus(double c): 현재 복소수객체와 형식파라메터실수객체를 호상 던 결과 역시 복소수값이고 그 메쏘드의 리용자에게 되돌려준다.

complexMulti(ComplexNumber c): 현재 복소수객체와 형식파라메터복소수객체를 곱한 결과 역시 복소수값이고 그 메쏘드의 리용자에게 되돌려준다.

complexMulti(double c): 현재 복소수객체와 형식파라메터실수객체를 곱한 결과 역시 복소수값이고 그 메쏘드의 리용자에게 되돌려준다.

toString( ): 현재 복소수객체의 실수부, 허수부를 a+bi문자렬형식으로 놓는다. 여기서 a, b는 각각 실수부와 허수부의 자료이다.

ComplexNumber클라스에서 객체지향의 어떤 기술을 리용하였는가? 이러한 기술을 리용하면 어떤 우점이 있는가?

ComplexNumber클라스는 객체지향의 내리적재기술을 리용하였으며 클라스내의 추상복소수의 연산을 실현하였다.

원천프로그람 ComplexNumber.java

```
public class ch4_e4_11
{
   public static void main(String args[ ])
   {
      ComplexNumber cNumber_1 = new ComplexNumber(3, -5);
      ComplexNumber cNumber_2 = new ComplexNumber(2, 2);
      Double d=10.0;
      System.out.println(cNumber_1.toString( ) + " 더하기 "
         + cNumber_2.toString( ) + " 같기 "
         +cNumber_1.complexAdd(cNumber_2).toString( ));
      System.out.println(cNumber_1.toString( ) + " 더하기 "
         + d + " 같기 " + cNumber_1.complexAdd(d).toString( ));
      System.out.println(cNumber_1.toString( ) + " 덜기 "
         + cNumber_2.toString( ) + " 같기"
         + cNumber_1.complexMinus(cNumber_2).toString( ));
      System.out.println(cNumber_1.toString( ) + " 덜기 "
         + d + " 같기 " + cNumber_1.complexMinus(d).toString( ));
      System.out.println(cNumber_1.toString( ) + " 곱하기 "
         + cNumber_2.toString( ) +" 같기 "
         + cNumber_1.complexMulti(cNumber_2).toString( ));
      System.out.println(cNumber_1.toString( ) + " 곱하기 "
         + d + " 같기 " + cNumber_1.complexMulti(d).toString( ));
   }
}
class ComplexNumber
{
   //마당
   private double m_dRealPart;
   private double m_dImaginPart;
   //구성자
   ComplexNumber( )
   {
      m_dRealPart=0.0;
      m_dImaginPart=0.0;
   }
   ComplexNumber(double r, double i)
   {
```

```
        m_dRealPart=r;
        m_dImaginPart=i;
    }
    ComplexNumber(ComplexNumber c)
    {
        m_dRealPart=c.getRealPart( );
        m_dImaginPart=c.getImaginaryPart( );
    }
    //get, set메쏘드
    double getRealPart( )
    {
        return m_dRealPart;
    }
    double getImaginaryPart( )
    {
        return m_dImaginPart;
    }
    void setRealPart(double d)
    {
        m_dRealPart=d;
    }
    void setImaginaryPart(double d)
    {
        m_dImaginPart=d;
    }
    //복소수연산메쏘드
    ComplexNumber complexAdd(ComplexNumber c)
    {
        return new ComplexNumber(this.m_dRealPart + c.getRealPart( ),
            this.m_dImaginPart + c.getImaginaryPart( ));
    }
    ComplexNumber complexAdd(double c)
    {
        return new ComplexNumber(this.m_dRealPart + c, this.m_dImaginPart);
    }
    ComplexNumber complexMinus(ComplexNumber c)
    {
        return new ComplexNumber(this.m_dRealPart - c.getRealPart( ),
            this.m_dImaginPart - c.getImaginaryPart( ));
```

```
        }
        ComplexNumber complexMinus(double c)
        {
            return new ComplexNumber(this.m_dRealPart - c, this.m_dImaginPart);
        }
        ComplexNumber complexMulti(ComplexNumber c)
        {
            return new ComplexNumber(this.m_dRealPart * c.getRealPart( )
                - this.m_dImaginPart * c.getImaginaryPart( ),
                this.m_dRealPart * c.getImaginaryPart( )
                + this.m_dImaginPart * c.getRealPart( ));
        }
        ComplexNumber complexMulti(double c)
        {
            return new ComplexNumber(this.m_dRealPart * c, this.m_dImaginPart * c);
        }
        //toString( )
        public String toString( )
        {
            return "(" + m_dRealPart + "+" + m_dImaginPart + "i" + ")";
        }
    }
```

**4.1.12. 문제 4.1.11의 복소수클라스를 리용하여 사용자가 입력한 복소수의 실수부와 허수부를 접수하고 복소수와 복소수, 복소수와 실수의 더하기, 덜기, 곱하기, 나누기를 계산하는 Applet프로그람을 작성하시오.**

여기서는 세가지 방안으로 실현한다. 첫째방안은 배렬을 리용하여 각종 도형대면성분을 조직하는것이고 두번째방안은 사용자정의클라스를 리용하여 도형대면의 입출력파일과 복소수자료를 내장하는것이며 세번째방안은 도형대면의 내리펼침차림표를 리용하여 대면을 간단히 하는것이다.

**원천프로그람1 ch4_e4_12a.java**

```
import java.applet. * ;
import java.awt. * ;
import java. awt.event. * ;
public class ch4_e4_12a extends Applet implements ActionListener
{
    final int NUMBER = 4;
    ComplexNumber cNumber_1;
```

```
ComplexNumber cNumber_2;
ComplexNumber cResult;
double d = 0.0;

TextField numComplexTfd[ ][ ], subComplexTfd[ ][ ];
Label resComplexLbl[ ][ ];
Button caculateBtn[ ];
final String COMMAND_NAME[ ] = {"ADD", "MINUS", "MULTIPLE", "DIVIDE"};

public ch4_e4_12a( )
{
   //초기화
   numComplexTfd = new TextField[NUMBER][2];
   subComplexTfd = new TextField[NUMBER][2];
   resComplexLbl = new Label[NUMBER][2];
   caculateBtn = new Button[NUMBER];

   for(int i = 0; i < NUMBER; i++)
   {
      for(int j = 0; j < 2; j++)
      {
         numComplexTfd[i][j] = new TextField(5);
         subComplexTfd[i][j] = new TextField(5);
         resComplexLbl[i][j] = new Label(" ");
      }
      caculateBtn[i] = new Button(" = ");
      caculateBtn[i].addActionListener(this);
      caculateBtn[i].setActionCommand(COMMAND_NAME[i]);
   }
   cNumber_1 = new ComplexNumber( );
   cNumber_2 = new ComplexNumber( );
   cResult = new ComplexNumber( );
   //매개 도형대면성분을 추가
   for(int i = 0; i < NUMBER; i++)
   {
      //첫번째 연산수의 입력대면성분
      add(new Label("("));
      add(numComplexTfd[i][0]);
      add(new Label("+"));
```

```
            add(numComplexTfd[i][1]);
            add(new Label("i"));

            //연산기호
            switch(i)
            {
               case 0: add(new Label("+"));
                  break;
               case 1: add(new Label("-"));
                  break;
               case 2: add(new Label("*"));
                  break;
               case 3: add(new Label("/"));
                  break;
            }

            //두번째 연산수의 입력대면성분
            add(new Label("("));
            add(subComplexTfd[i][0]);
            add(new Label("+"));
            add(subComplexTfd[i][1]);
            add(new Label("i )"));

            //연산시작단추
            add(caculateBtn[i]);

            //연산결과의 출력대면성분
            add(new Label("("));
            add(resComplexLbl[i][0]);
            add(new Label("+"));
            add(resComplexLbl[i][1]);
            add(new Label("i )"));
         } //for
      } //구성자

      public void init( )
      {
         resetTextField( );
      }
```

```
public void actionPerformed(ActionEvent e)
{
   int flag = 0;

   try
   {
      // 어떤 종류의 연산인가를 확정
      if(e.getActionCommand( ).equals(COMMAND_NAME[0]))
         flag = 0;
      else if(e.getActionCommand( ).equals(COMMAND_NAME[1]))
         flag = 1;
      else if(e.getActionCommand( ).equals(COMMAND_NAME[2]))
         flag = 2;
      else if(e.getActionCommand( ).equals(COMMAND_NAME[3]))
         flag = 3;
      else
      {
         showStatus("이 사건의 처리조작을 아직 정의하지 않았다."
            +e.toString( ));
         return;
      }

      // 첫번째 연산수의 수값을 얻기
      cNumber_1.setRealPart(
      Double.parseDouble(numComplexTfd[flag][0].getText( )));
      cNumber_1.setImaginaryPart(
         Double.parseDouble(numComplexTfd[flag][1].getText( )));
      // 두번째 연산수가 실수일 때 대응하는 계산을 완성
      if(subComplexTfd[flag][1].getText( ).equals(" "))
      {
         d = Double.parseDouble(subComplexTfd[flag][0].getText( ));
         switch(flag)
         {
            case 0: cResult = cNumber_1.complexAdd(d);
               break;
            case 1: cResult = cNumber_1.complexMinus(d);
               break;
            case 2: cResult = cNumber_1.complexMulti(d);
               break;
```

```
            case 3: cResult = cNumber_1.complexDivide(d);
                break;
        }
    }
    // 두번째 연산수가 복소수이라면 대응하는 계산을 완성
    else
    {
        cNumber_2.setRealPart
          (Double.parseDouble(subComplexTfd[flag][0].getText( )));
        cNumber_2.setImaginaryPart
          (Double.parseDouble(subComplexTfd[flag][1].getText( )));
        switch(flag)
        {
            case 0: cResult = cNumber_1.complexAdd(cNumber_2);
                break;
            case 1: cResult = cNumber_1.complexMinus(cNumber_2);
                break;
            case 2: cResult = cNumber_1.complexMulti(cNumber_2);
                break;
            case 3: cResult = cNumber_1.complexDivide(cNumber_2);
                break;
        }
    }
    //연산결과를 현시
    resComplexLbl[flag][0].setText(Double.toString(cResult.getRealPart( )));
    resComplexLbl[flag][1].setText(
        Double.toString(cResult.getImaginaryPart( )));
  }
  catch(NumberFormatException nfe)
  {
    showStatus("자료형식이 틀리므로 다시 입력하시오.");
    resetTextField( );
  }
}
// 입출력마당을 비게 한다.
void resetTextField( )
{
  for(int i = 0; i  <NUMBER; i++)
  {
    for(int j = 0; j < 2; j++)
```

```
                {
                    numComplexTfd[i][j].setText(" ");
                    subComplexTfd[i][j].setText(" ");
                    resComplexLbl[i][j].setText(" ");
                }
            }
        }
    }
    class ComplexNumber
    {
        // 마당
        private double m_dRealPart;
        private double m_dImaginPart;
        // 구성자
        ComplexNumber( )
        {
            m_dRealPart = 0.0;
            m_dImaginPart = 0.0;
        }
        ComplexNumber(double r,double i)
        {
            m_dRealPart = r;
            m_dImaginPart = i;
        }
        ComplexNumber(ComplexNumber c)
        {
            m_dRealPart = c.getRealPart( );
            m_dImaginPart = c.getImaginaryPart( );
        }

        // get, set메쏘드
        double getRealPart( )
        {
            return m_dRealPart;
        }
        double getImaginaryPart( )
        {
            return m_dImaginPart;
        }
```

```
void setRealPart(double d)
{
   m_dRealPart = d;
}
void setImaginaryPart(double d)
{
   m_dImaginPart = d;
}
// 복소수의 공액
ComplexNumber ComplexConjugate( )
{
   return new ComplexNumber(m_dRealPart, -1 * m_dImaginPart);
}
// 복소수연산메쏘드
ComplexNumber complexAdd(ComplexNumber c)
{
   return new ComplexNumber(this.m_dRealPart + c.getRealPart( ),
      this.m_dImaginPart + c.getImaginaryPart( ));
}
ComplexNumber complexAdd(double c)
{
   return new ComplexNumber(this.m_dRealPart + c, this.m_dImaginPart);
}
ComplexNumber complexMinus(ComplexNumber c)
{
   return new ComplexNumber(this.m_dRealPart - c.getRealPart( ),
   this.m_dImaginPart - c.getImaginaryPart( ));
}
ComplexNumber complexMinus(double c)
{
   return new ComplexNumber(this.m_dRealPart - c,this.m_dImaginPart);
}
ComplexNumber complexMulti(ComplexNumber c)
{
   return new ComplexNumber(this.m_dRealPart * c.getRealPart( )
      - this.m_dImaginPart * c.getImaginaryPart( ),
      this.m_dRealPart * c.getImaginaryPart( )
      + this.m_dImaginPart * c.getRealPart( ));
}
ComplexNumber complexMulti(double c)
```

```
    {
        return new ComplexNumber(this.m_dRealPart * c, this.m_dImaginPart * c);
    }
    ComplexNumber complexDivide(double d)
    {
        return new ComplexNumber(this.m_dRealPart / d, this.m_dImaginPart / d);
    }
    ComplexNumber complexDivide(ComplexNumber c)
    {
        double mod = c.getRealPart( ) * c.getRealPart( )
            + c.getImaginaryPart( ) * c.getImaginaryPart( );
        if(mod == 0)
            return new ComplexNumber(0, 0);
        else
            return this.complexMulti(c.ComplexConjugate( )).complexDivide(mod);
    }
    // toString( )
    public String toString( )
    {
        return "(" + m_dRealPart + "+" + m_dImaginPart + "i" + ")";
    }
}
```

원천프로그람2 ch4_e4_12b.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class ch4_e4_12b extends Applet implements ActionListener
{
    final int NUMBER = 4;
    ComplexInputPanel inputComplexPnl[ ][ ];
    ComplexOutputPanel outputComplexPnl[ ];
    Button caculateBtn[ ];
    final String COMMAND_NAME[ ] = {"ADD", "MINUS", "MULTIPLE", "DIVIDE"};
    public ch4_e4_12b( )
    {
        super( );
```

```
//초기화
inputComplexPnl = new ComplexInputPanel[NUMBER][2];
outputComplexPnl = new ComplexOutputPanel[NUMBER];
caculateBtn = new Button[NUMBER];
for(int i =0; i < NUMBER; i++)
{
   for(int j = 0; j < 2; j++)
      inputComplexPnl[i][j] = new ComplexInputPanel( );
   outputComplexPnl[i] = new ComplexOutputPanel( );
   caculateBtn[i] = new Button("=");
   caculateBtn[i].addActionListener(this);
   caculateBtn[i].setActionCommand(COMMAND_NAME[i]);
}

//매개 도형대면성분을 추가
for(int i = 0; i < NUMBER; i++)
{
   //첫번째 연산수의 입력대면성분
   add(inputComplexPnl[i][0]);
   //연산기호
   switch(i)
   {
      case 0: add(new Label("+"));
         break;
      case 1: add(new Label("-"));
         break;
      case 2: add(new Label("*"));
         break;
      case 3: add(new Label("/"));
         break;
   }

   //두번째 연산수의 입력대면성분
   add(inputComplexPnl[i][1]);
   //연산시작단추
   add(caculateBtn[i]);

   //연산결과의 출력대면성분
   add(outputComplexPnl[i]);
```

```
        } //for
    }
    public void init( )
    {
        for(int i = 0; i < NUMBER; i++)
        {
            for(int j = 0; j < 2; j++)
                inputComplexPnl[i][j].resetFaces( );
            outputComplexPnl[i].resetFaces( );
        }
    } //init
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            if(e.getActionCommand( ).equals(COMMAND_NAME[0]))
            {
                inputComplexPnl[0][0].getComplexNumber( );
                outputComplexPnl[0].setComplexNumber(inputComplexPnl[0][0]
                    .complexAdd(inputComplexPnl[0][1].getComplexNumber( )));
            }
            else if(e.getActionCommand( ).equals(COMMAND_NAME[1]))
            {
                inputComplexPnl[1][0].getComplexNumber( );
                outputComplexPnl[1].setComplexNumber(inputComplexPnl[1][0]
                    .complexMinus(inputComplexPnl[1][1].getComplexNumber( )));
            }
            else if(e.getActionCommand( ).equals(COMMAND_NAME[2]))
            {
                inputComplexPnl[2][0].getComplexNumber( );
                outputComplexPnl[2].setComplexNumber(inputComplexPnl[2][0]
                    .complexMulti(inputComplexPnl[2][1].getComplexNumber( )));
            }
            else if(e.getActionCommand( ).equals(COMMAND_NAME[3]))
            {
                inputComplexPnl[3][0].getComplexNumber( );
                outputComplexPnl[3].setComplexNumber(inputComplexPnl[3][0]
                    .complexDivide(inputComplexPnl[3][1].getComplexNumber( )));
            }
```

```
            else
            {
                showStatus("이 사건의 처리조작이 정의되지 않았다."+ e.toString( ));
                return;
            }
        }
        catch(NumberFormatException nfe)
        {
            showStatus("자료형식이 틀리므로 다시 입력하시오.");
            for(int i = 0; i < NUMBER; i++)
            {
                for(int j = 0; j < 2; j++)
                    inputComplexPnl[i][j].resetFaces( );
                outputComplexPnl[i].resetFaces( );
            }
        }
    }
}
interface ComplexCalculationsInterface
{
    public ComplexNumber complexAdd(ComplexNumber c);
    public ComplexNumber complexAdd(double d);
    public ComplexNumber complexMinus(ComplexNumber c);
    public ComplexNumber complexMinus(double d);
    public ComplexNumber complexMulti(ComplexNumber c);
    public ComplexNumber complexMulti(double d);
    public ComplexNumber complexDivide(double d);
    public ComplexNumber complexDivide(ComplexNumber c);
}
abstract class ComplexPanel extends Panel implements
    ComplexCalculationsInterface
{
    ComplexNumber innerComplex;
    Label prefixLabel, midLabel, suffixLabel;
    ComplexPanel( )
    {
        super( );
        innerComplex = new ComplexNumber( );
        prefixLabel = new Label("(");
        midLabel = new Label("+");
```

```
        suffixLabel = new Label("i )");
        createAddFaces( );
    }
    abstract void createAddFaces( );
    abstract ComplexNumber getComplexNumber( ) throws NumberFormatException;
    abstract void setComplexNumber(ComplexNumber c);
    abstract void resetFaces( );
    public ComplexNumber complexAdd(ComplexNumber c)
    {
        return innerComplex.complexAdd(c);
    }
    public ComplexNumber complexAdd(double d)
    {
        return innerComplex.complexAdd(d);
    }
    public ComplexNumber complexMinus(ComplexNumber c)
    {
        return innerComplex.complexMinus(c);
    }
    public ComplexNumber complexMinus(double d)
    {
        return innerComplex.complexMinus(d);
    }
    public ComplexNumber complexMulti(ComplexNumber c)
    {
        return innerComplex.complexMulti(c);
    }
    public ComplexNumber complexMulti(double d)
    {
        return innerComplex.complexMulti(d);
    }
    public ComplexNumber complexDivide(double d)
    {
        return innerComplex.complexDivide(d);
    }
    public ComplexNumber complexDivide(ComplexNumber c)
    {
        return innerComplex.complexDivide(c);
    }
}
```

```
class ComplexInputPanel extends ComplexPanel
{
   private TextField realTfd, imaginaryTfd;
   void createAddFaces( )
   {
      realTfd = new TextField(3);
      imaginaryTfd = new TextField(3);
      add(prefixLabel);
      add(realTfd);
      add(midLabel);
      add(imaginaryTfd);
      add(suffixLabel);
   }
   ComplexNumber getComplexNumber( ) throws NumberFormatException
   {
      innerComplex = new ComplexNumber(Double.parseDouble(realTfd.getText( )),
         Double.parseDouble((imaginaryTfd.getText( ).equals(" "))? "0":
         imaginaryTfd.getText( )));
      return innerComplex;
   }
   void setComplexNumber(ComplexNumber c)
   {
      double r, i;
      r = c.getRealPart( );
      i = c.getImaginaryPart( );
      innerComplex.setRealPart(r);
      innerComplex.setImaginaryPart(i);
      realTfd.setText(Double.toString(r));
      imaginaryTfd.setText(Double.toString(i));
   }
   void resetFaces( )
   {
      realTfd.setText(" ");
      imaginaryTfd.setText(" ");
   }
}
class ComplexOutputPanel extends ComplexPanel
{
   private Label realLbl, imaginaryLbl;
   void createAddFaces( )
```

```
    {
        realLbl = new Label(" ");
        imaginaryLbl = new Label(" ");
        add(prefixLabel);
        add(realLbl);
        add(midLabel);
        add(imaginaryLbl);
        add(suffixLabel);
    }
    ComplexNumber getComplexNumber( ) throws NumberFormatException
    {
        innerComplex = new ComplexNumber(Double.parseDouble(realLbl.getText( )),
            Double.parseDouble((imaginaryLbl.getText( ).equals(" "))? "0":
            imaginaryLbl.getText( )));
        return innerComplex;
    }
    void setComplexNumber(ComplexNumber c)
    {
        double r, i;
        r = c.getRealPart( );
        i = c.getImaginaryPart( );
        innerComplex.setRealPart(r);
        innerComplex.setImaginaryPart(i);
        realLbl.setText(Double.toString(r));
        imaginaryLbl.setText(Double.toString(i));
    }
    void resetFaces( )
    {
        realLbl.setText(" ");
        imaginaryLbl.setText(" ");
    }
}
class ComplexNumber implements ComplexCalculationsInterface
{
    //마당
    private double m_dRealPart;
    private double m_dImaginPart;
    //구성자
    ComplexNumber( )
```

```
{
    m_dRealPart = 0.0;
    m_dImaginPart = 0.0;
}
ComplexNumber(double r, double i)
{
    m_dRealPart = r;
    m_dImaginPart = i;
}
ComplexNumber(ComplexNumber c)
{
    m_dRealPart = c.getRealPart( );
    m_dImaginPart = c.getImaginaryPart( );
}
//get, set메쏘드
double getRealPart( )
{
    return m_dRealPart;
}
double getImaginaryPart( )
{
    return m_dImaginPart;
}
void setRealPart(double d)
{
    m_dRealPart = d;
}
void setImaginaryPart(double d)
{
    m_dImaginPart = d;
}
//복소수의 공액
ComplexNumber complexConjugate( )
{
    return new ComplexNumber(m_dRealPart, -1 * m_dImaginPart);
}
//복소수연산메쏘드
public ComplexNumber complexAdd(ComplexNumber c)
{
    return new ComplexNumber(this.m_dRealPart + c.getRealPart( ),
```

```
        this.m_dImaginPart + c.getImaginaryPart( ));
}
public ComplexNumber complexAdd(double c)
{
    return new ComplexNumber(this.m_dRealPart+c, this.m_dImaginPart);
}
public ComplexNumber complexMinus(ComplexNumber c)
{
    return new ComplexNumber(this.m_dRealPart - c.getRealPart( ),
        this.m_dImaginPart - c.getImaginaryPart( ));
}
public ComplexNumber complexMinus(double c)
{
    return new ComplexNumber(this.m_dRealPart - c, this.m_dImaginPart);
}
public ComplexNumber complexMulti(ComplexNumber c)
{
    return new ComplexNumber(this.m_dRealPart * c.getRealPart( )
        - this.m_dImaginPart * c.getImaginaryPart( ),
        this.m_dRealPart * c.getImaginaryPart( )
        + this.m_dImaginPart * c.getRealPart( ));
}
public ComplexNumber complexMulti(double c)
{
    return new ComplexNumber(this.m_dRealPart * c, this.m_dImaginPart * c);
}
public ComplexNumber complexDivide(double d)
{
    return new ComplexNumber(this.m_dRealPart / d, this.m_dImaginPart / d);
}
public ComplexNumber complexDivide(ComplexNumber c)
{
    double mod = c.getRealPart( ) * c.getRealPart( )
        + c.getImaginaryPart( ) * c.getImaginaryPart( );
    if(mod == 0)
        return new ComplexNumber(0, 0);
    else
        return this.complexMulti(c.complexConjugate( )).complexDivide(mod);
}
```

```
    //toString( )
    public String toString( )
    {
        return "(" + m_dRealPart + "+" + m_dImaginPart + "i " + ")";
    }
}
```

원천프로그람3 ch4_e4_12c.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch4_e4_12c extends Applet implements ItemListener
{
    ComplexInputPanel firstInputComplexPnl, sndInputComplexPnl;
    ComplexOutputPanel outputComplexPnl;
    Choice operators;
    public ch4_e4_12c( )
    {
        super( );
        //초기화
        firstInputComplexPnl = new ComplexInputPanel( );
        sndInputComplexPnl = new ComplexInputPanel( );
        outputComplexPnl = new ComplexOutputPanel( );
        operators = new Choice( );
        operators.add(" ");
        operators.add("+");
        operators.add("-");
        operators.add("*");
        operators.add("/");
        operators.addItemListener(this);
        //매개 도형대면성분 추가
        add(firstInputComplexPnl);
        add(operators);
        add(sndInputComplexPnl);
        add(new Label("="));
        add(outputComplexPnl);
    }
```

```
public void init( )
 {
    firstInputComplexPnl.resetFaces( );
    sndInputComplexPnl.resetFaces( );
    outputComplexPnl.resetFaces( );
 }
 public void itemStateChanged(ItemEvent e)
 {
    Choice temp;
    try
    {
       if(e.getItemSelectable( ) instanceof Choice)
       {
          temp = (Choice)(e.getItemSelectable( ));
          switch(temp.getSelectedItem( ).charAt(0))
          {
             case '+':
                firstInputComplexPnl.getComplexNumber( );
                outputComplexPnl.setComplexNumber(
                   firstInputComplexPnl.complexAdd(
                   sndInputComplexPnl.getComplexNumber( )));
                break;
             case '-':
                firstInputComplexPnl.getComplexNumber( );
                outputComplexPnl.setComplexNumber(
                   firstInputComplexPnl.complexMinus(
                   sndInputComplexPnl.getComplexNumber( )));
                break;
             case '*':
                firstInputComplexPnl.getComplexNumber( );
                outputComplexPnl.setComplexNumber(
                   firstInputComplexPnl.complexMulti(
                   sndInputComplexPnl.getComplexNumber( )));
                break;
             case '/':
                firstInputComplexPnl.getComplexNumber( );
                outputComplexPnl.setComplexNumber(
                   firstInputComplexPnl.complexDivide(
                   sndInputComplexPnl.getComplexNumber( )));
```

```
                break;
            default:
            }
        }
        else
        {
            showStatus("이 사건의 처리조작을 아직 정의하지 않았다." + e.toString( ));
            return;
        }
    }
    catch(NumberFormatException nfe)
    {
        showStatus("자료형식이 틀리면 다시 입력하시오.");
        firstInputComplexPnl.resetFaces( );
        sndInputComplexPnl.resetFaces( );
        outputComplexPnl.resetFaces( );
    }
  }
}
interface ComplexCalculationsInterface
{
    public ComplexNumber complexAdd(ComplexNumber c);
    public ComplexNumber complexAdd(double d);
    public ComplexNumber complexMinus(ComplexNumber c);
    public ComplexNumber complexMinus(double d);
    public ComplexNumber complexMulti(ComplexNumber c);
    public ComplexNumber complexMulti(double d);
    public ComplexNumber complexDivide(double d);
    public ComplexNumber complexDivide(ComplexNumber c);
}
abstract class ComplexPanel extends Panel implements
    ComplexCalculationsInterface
{
    ComplexNumber innerComplex;
    Label prefixLabel, midLabel, suffixLabel;
    ComplexPanel( )
    {
        super( );
        innerComplex = new ComplexNumber( );
        prefixLabel = new Label("(");
```

```
        midLabel = new Label("+");
        suffixLabel = new Label("i");
        createAddFaces( );
    }
    abstract void createAddFaces( );
    abstract ComplexNumber getComplexNumber( ) throws NumberFormatException;
    abstract void setComplexNumber(ComplexNumber c);
    abstract void resetFaces( );
    public ComplexNumber complexAdd(ComplexNumber c)
    {
        return innerComplex.complexAdd(c);
    }
    public ComplexNumber complexAdd(double d)
    {
        return innerComplex.complexAdd(d);
    }
    public ComplexNumber complexMinus(ComplexNumber c)
    {
        return innerComplex.complexMinus(c);
    }
    public ComplexNumber complexMinus(double d)
    {
        return innerComplex.complexMinus(d);
    }
    public ComplexNumber complexMulti(ComplexNumber c)
    {
        return innerComplex.complexMulti(c);
    }
    public ComplexNumber complexMulti(double d)
    {
        return innerComplex.complexMulti(d);
    }
    public ComplexNumber complexDivide(double d)
    {
        return innerComplex.complexDivide(d);
    }
    public ComplexNumber complexDivide(ComplexNumber c)
    {
        return innerComplex.complexDivide(c);
```

```
    }
}
class ComplexInputPanel extends ComplexPanel
{
    private TextField realTfd, imaginaryTfd;
    void createAddFaces( )
    {
        realTfd = new TextField(3);
        imaginaryTfd = new TextField(3);
        add(prefixLabel);
        add(realTfd);
        add(midLabel);
        add(imaginaryTfd);
        add(suffixLabel);
    }
    ComplexNumber getComplexNumber( ) throws NumberFormatException
    {
        innerComplex = new ComplexNumber(
        Double.parseDouble(realTfd.getText( )),
        Double.parseDouble((imaginaryTfd.getText( ).equals(" "))? "0":
            imaginaryTfd.getText( )));
        return innerComplex;
    }
    void setComplexNumber(ComplexNumber c)
    {
        double r, i;
        r = c.getRealPart( );
        i = c.getImaginaryPart( );
        innerComplex.setRealPart(r);
        innerComplex.setImaginaryPart(i);
        realTfd.setText(Double.toString(r));
        imaginaryTfd.setText(Double.toString(i));
    }
    void resetFaces( )
    {
        realTfd.setText(" ");
        imaginaryTfd.setText(" ");
    }
}
```

```
class ComplexOutputPanel extends ComplexPanel
{
   private Label realLbl,imaginaryLbl;
   void createAddFaces( )
   {
      realLbl = new Label(" ");
      imaginaryLbl = new Label(" ");
      add(prefixLabel);
      add(realLbl);
      add(midLabel);
      add(imaginaryLbl);
      add(suffixLabel);
   }
   ComplexNumber getComplexNumber( ) throws NumberFormatException
   {
      innerComplex = new ComplexNumber(Double.parseDouble(realLbl.getText( )),
         Double.parseDouble((imaginaryLbl.getText( ).equals(" "))? "0":
         imaginaryLbl.getText( )));
      return innerComplex;
   }
   void setComplexNumber(ComplexNumber c)
   {
      double r, i;
      r = c.getRealPart( );
      i = c.getImaginaryPart( );
      innerComplex.setRealPart(r);
      innerComplex.setImaginaryPart(i);
      realLbl.setText(Double.toString(r));
      imaginaryLbl.setText(Double.toString(i));
   }
   void resetFaces( )
   {
      realLbl.setText(" ");
      imaginaryLbl.setText(" ");
   }
}
class ComplexNumber implements ComplexCalculationsInterface
{
   //마당
```



교육성 교육정보쎈터

```
private double m_dRealPart;
private double m_dImaginPart;
//구성자
ComplexNumber( )
{
   m_dRealPart = 0.0;
   m_dImaginPart = 0.0;
}
ComplexNumber(double r, double i)
{
   m_dRealPart = r;
   m_dImaginPart = i;
}
ComplexNumber(ComplexNumber c)
{
   m_dRealPart = c.getRealPart( );
   m_dImaginPart = c.getImaginaryPart( );
}
//get, set메쏘드
double getRealPart( )
{
   return m_dRealPart;
}
double getImaginaryPart( )
{
   return m_dImaginPart;
}
void setRealPart(double d)
{
   m_dRealPart = d;
}
void setImaginaryPart(double d)
{
   m_dImaginPart = d;
}
//복소수의 공액
ComplexNumber complexConjugate( )
{
   return new ComplexNumber(m_dRealPart, -1 * m_dImaginPart);
}
```

```
//복소수의 연산메쏘드
public ComplexNumber complexAdd(ComplexNumber c)
{
    return new ComplexNumber(this.m_dRealPart + c.getRealPart( ),
        this.m_dImaginPart + c.getImaginaryPart( ));
}
public ComplexNumber complexAdd(double c)
{
    return new ComplexNumber(this.m_dRealPart + c, this.m_dImaginPart);
}
public ComplexNumber complexMinus(ComplexNumber c)
{
    return new ComplexNumber(this.m_dRealPart - c.getRealPart( ),
        this.m_dImaginPart - c.getImaginaryPart( ));
}
public ComplexNumber complexMinus(double c)
{
    return new ComplexNumber(this.m_dRealPart - c, this.m_dImaginPart);
}
public ComplexNumber complexMulti(ComplexNumber c)
{
    return new ComplexNumber(this.m_dRealPart * c.getRealPart( )
        - this.m_dImaginPart * c.getImaginaryPart( ),
        this.m_dRealPart * c.getImaginaryPart( )
        + this.m_dImaginPart * c.getRealPart( ));
}
public ComplexNumber complexMulti(double c)
{
    return new ComplexNumber(this.m_dRealPart * c, this.m_dImaginPart * c);
}
public ComplexNumber complexDivide(double d)
{
    return new ComplexNumber(this.m_dRealPart / d, this.m_dImaginPart / d);
}
public ComplexNumber complexDivide(ComplexNumber c)
{
    double mod = c.getRealPart( ) * c.getRealPart( )
        + c.getImaginaryPart( ) * c.getImaginaryPart( );
    if(mod == 0)
```

```
            return new ComplexNumber(0, 0);
        else
            return this.complexMulti(c.complexConjugate( )).complexDivide(mod);
    }
    //toString( )
    public String toString( )
    {
        return "(" + m_dRealPart + "+" + m_dImaginPart + "i" + ")";
    }
}
```

### 4.1.13. 구성자를 내리적재할수 있는가? 실례를 드시오.

구성자를 내리적재할수 있다. 실례를 들면 다음과 같다.

```
원천프로그람 ch4_e4_13.java
public class ch4_e4_13
{
    public static void main(String args[ ])
    {
        System.out.println("상위클라스객체만들기:");
        SuperClass sc0 = new SuperClass( );
        System.out.println("\n첫번째 하위클라스객체만들기:");
        SubClass sc1 = new SubClass( );
        System.out.println("\n두번째 하위클라스객체만들기:");
        SubClass sc2 = new SubClass(1);
    }
}
class SuperClass
{
    SuperClass( )
    {
        System.out.println("상위클라스의 구성자");
    }
}
class SubClass extends SuperClass
{
    SubClass( )
    {
        System.out.println("하위클라스의 첫번째구성자");
    }
```

```
        SubClass(int i)
        {
           System.out.println("하위클라스의 두번째구성자");
        }
    }
```

**4.1.14. 패키지란 무엇이며 어떤 작용을 하는가?**

패키지란 클라스의 모임이다. 패키지의 작용은 주로 협동작업을 요구하는 서로 다른 클라스를 함께 조직하여 프로그람의 기능과 구조를 명백하게 하고 클라스들을 보다 효과적으로 관리하는데 있다.

**4.1.15. 패키지를 어떻게 만들며 어떤 경우에 프로그람안에 패키지를 만들어야 하는가?**

Java프로그람에서는 예약어 package를 리용하여 패키지를 만든다. 만일 프로그람에서 정의한 몇개의 클라스가 하나의 작은 집단을 만들며 이것이 기타 프로그람에 의하여 인용될수 있다면 항상 프로그람안에서 패키지를 만들어 리용할수 있다.

**4.1.16. 패키지의 임의의 클라스를 어떻게 인용할수 있는가? 전체 패키지를 인용하자면 어떻게 하여야 하는가? 만일 Java Applet프로그람을 작성할 때 java.applet패키지의 일부를 적재하려면 어떻게 하여야 하는가?**

패키지의 어떤 클라스를 인용하려면 예약어 import와 사용클라스이름외에 패키지이름을 리용하여 앞붙이를 더 만들어 주어야 한다.

실례로 :

```
import java.awt.Button;
```

전체 패키지를 인용하는것은 패키지의 모든 클라스를 인용한다는것이며 형식은 다음과 같다.

```
import java.awt.*;
```

또한 Applet클라스를 적재하려면 명령을 다음과 같이 쓰면 된다.

```
import java.applet.Applet;
```

**4.1.17. CLASSPATH는 어떤 환경변수인가? 그것이 프로그람의 실행에 어떻게 영향을 주는가? 이 환경변수를 어떻게 설정하고 수정하는가?**

CLASSPATH는 기정의 패키지파일(바이트코드파일)경로를 지정하는 환경변수이다.

만일 CLASSPATH가 정확하게 설정되지 않았다면 프로그람을 번역하고 실행할 때 요구하는 클라스의 바이트코드파일을 찾을수 없으므로 프로그람의 정상실행을 할수 없게 한다.

아래의 명령을 리용하여 CLASSPATH를 다음과 같이 설정 또는 수정할수 있다.

```
SET CLASSPATH=.;C:\
```

**4.1.18. 대면(interface)이란 무엇이며 왜 대면을 정의해야 하는가? 대면과 클라스는 어떤 차이점이 있는가?**

대면은 클라스들사이에 다중계승기능을 실현하는데 리용한다. 대면과 클라스는 비슷하지만 대면은 오직 상수와 추상메쏘드만을 포함할수 있다.

대면을 정의하는것은 하나의 추상기능과 속성모임을 정의하는것과 류사하므로 Java프로그람의 클라스의 층구조가 보다 합리적으로 되도록 하고 다중계승을 실현할수 있게 한다.

**4.1.19. 대면은 어떤 예약어를 리용하여 정의하는가?**

Java프로그람은 예약어 interface를 리용하여 대면을 정의하며 그 형식은 다음과 같다.

interface 대면이름;

**4.1.20. 클라스는 어떻게 대면을 실현하는가? 어떤 대면을 실현하는 클라스가 반드시 그 대면의 모든 추상메쏘드를 내리적재하여야 하는가?**

예약어 implements를 리용하여 클라스가 어느 대면을 실현하는가를 설정한다.

대면을 실현하는 클라스가 추상클라스가 아니라면 내리적재를 통하여 그 대면의 모든 추상메쏘드를 실현하여야 한다.

만일 이 클라스가 추상클라스라면 그 대면의 모든 추상메쏘드를 반드시 실현할 필요가 없다.

## 4.2. 보충문제

4.2.1. 추상클라스와 대면은 어떤 차이점을 가지는가?

4.2.2. 아래의 개념을 계승관계나무로 만드시오.

중어, 자연언어, Java언어, C언어, 언어, 프로그람작성언어, 발음언어, 조선어

4.2.3. Java언어의 +기호도 다형성을 가진다. 즉 산수더하기와 문자렬더하기를 실현할수 있다. 이 말이 정확한가?

4.2.4. 아래의 코드에 틀린것이 있는가? 만일 틀린것이 있다면 어떻게 고쳐야 하는가?

```
Class supClass
{
   int supVariable = 0;
   supClass(int a)
   {
      supVariable = a;
   }
}
class supClass extends supClass
{
   int subVariable = 0;
   subClass(int a)
   {
      subVariable = a;
   }
}
```

4.2.5. 다음과 같이 정의된 클라스가 있다고 하자.

```
Class supClass
{
   int supVariable = 0;
   supClass(int a)
   {
      supVriable = a;
   }
}
class subClass extends supClass
{
   int subVariable = 0;
   subclass(int a)
   {
```

```
            super(a);
            subVariable = a;
        }
    }
```

아래의 문장이 틀리는가? 만일 틀린다면 어떻게 고쳐야 하는가?

```
    Subclass subc = new supClass(3);
```

4.2.6. 대면은 계승될수 있는가? 아래의 코드를 읽고 SubInterface의 추상메쏘드를 쓰시오.

```
    interface SupInterface
    {
        public abstract int supMethod( );
    }
    interface SubInterface extends SupInterface
    {
        public abstract string subMethod( );
    }
```

# 제5장. 도구클라스

## 5.1. 련습

**5.1.1. Java체계클라스에서 Object클라스는 어떤 특수한 점이 있는가? 어떤 조건에서 리용하는가?**

Object클라스는 모든 Java클라스, 체계클라스 혹은 사용자정의클라스의 직접 또는 간접상위클라스이다. 임의의 Java객체를 Object클라스의 객체로 볼수 있기때문에 Object는 넓은 조건에서 리용할수 있다.

실례로 어떤 형식파라메터의 형으로 리용된다.

**5.1.2. 자주 리용하는 자료형클라스에는 어떤것이 있으며 자료형클라스와 기본자료형이 어떤 관계가 있는가를 말하시오.**

자주 리용하는 자료형클라스에는 Boolean, Byte, Character, Double, Float, Integer, Long, Short 등이 있다.

자료형클라스는 기본자료형에 기초하고있으며 거기에 기본자료형의 마당 및 그와 관계되는 조작이 내장되여있다.

**5.1.3. Math클라스는 어떤 기능을 수행하는데 리용하는가? x, y가 옹근수형변수이고 d가 배정밀도변수일 때 아래의 조작을 수행하는 표현식을 쓰시오.**

1) x의 y제곱을 구하시오.

2) x와 y의 최소값을 구하시오.

3) d의 옹근수부를 취한 결과를 구하시오.

4) d의 반올림결과를 구하시오.

5) atan(d)의 수값을 구하시오.

Math클라스에는 자주 리용되는 몇개의 수학연산이 포함되여있다.

1) Math.pow(x, y)

2) Math.min(x, y)

3) Math.floor(d)

4) Math.round(d)

5) Math.atan(d)

**5.1.4.** Math.random( )**메쏘드는 어떤 기능을 실현하는데 리용되는가? 아래의 문장은 어떤 작용을 하는가?**

(int)(Math.random( ) * 6) + 1

1~6사이에서 100개의 우연수를 발생하는 프로그람을 작성하고 1~6사이의 매개 수가 나타날 확률을 구하시오. 1000개의 우연수를 발생하며 그 확률을 구하는 프로그람으로 고치시오. 두 결과를 비교하여 결론을 얻어내시오.

Math.random( )메쏘드는 0과 1(1을 포함하지 않는다.)사이의 란수를 발생하는데 리용된다.

명령 (int)(Math.random( )*6)+1은 1~6사이의 란수를 발생하는데 리용된다.

```
원천프로그람 ch5_e5_4.java
public class ch5_e5_4
{
   public static void main(String args[ ])
   {
      final int NUMBER = 1000;
      int count = 10;
      int randomNum = 0;
      int probability[ ] = new int[6];

      for(int i = 0; i < 6; i++)
      {
         probability[i] = 0;
      }
      for(int i = 0; i < NUMBER; i++)
      {
         randomNum = (int)(Math.random( ) * 6) + 1;
         probability[randomNum -1]++;
         System.out.print(randomNum + "\t");
         if(1 % count == 9)
         System.out.println( );
      }
      for(int i = 0; i < 6; i++)
      {
         System.out.println("\n" + (i + 1) + ":\t" + probability[i]);
      }
   }
}
```

Number를 고치면 발생하는 우연수의 개수를 변화시킬수 있다.

**5.1.5. System.exit( )메쏘드는 어떤 작용을 하는가? 어떤 경우에 그것을 리용하는가?**

System.exit( )메쏘드는 Java가상기계를 강제조작하여 실행상태로부터 탈퇴하게 하며 조작체계로 하여금 상태정보를 되돌려주게 한다.

일반적으로 프로그람이 체계오유(실례로 입출력오유)를 검색할 때 이 메쏘드를 리용한다.

**5.1.6. Applet의 기본작업원리를 서술하시오. 체계클라스 Applet는 클라스서고의 어느 패키지에 속하는가?**

체계클라스 Applet는 java.applet패키지에 속한다. Applet는 패키지 혹은 조종파일과 류사한 하나의 특수한 클라스이다. 작업원리는 다음과 같다.

번역된 Applet클라스의 코드파일(.class파일)은 특정한 WWW봉사기에 보관되며 동시에 같은 혹은 다른 WWW봉사기에 그 코드파일이름이 삽입되여있는 HTML파일이 보관된다. 어떤 열람기가 봉사기에 Applet를 삽입한 HTML파일을 내리적재할것을 요구할 때 그 HTML파일은 WWW봉사기로부터 사용자말단까지 내리적재되고 WWW열람기에 의하여 HTML의 여러가지 꼬리표가 해석되며 약속에 따라 파일의 정보를 일정한 형식으로 사용자화면에 현시한다.

열람기가 HTML파일의 특수꼬리표와 맞다들리면 그것에 Applet의 코드화일이 삽입되여있다는것으로 인식하고 이 Applet의 이름과 위치에 근거하여 코드를 WWW봉사기로부터 자기한테까지 자동적으로 내리적재하며 열람기자체가 가지고있는 Java해석기를 리용하여 그 코드를 직접 실행한다.

**5.1.7. Applet클라스에 있는 열람기에 의하여 자동적으로 리용될수 있는 메쏘드들의 작용을 간단히 말해보시오.**

Applet클라스에서 열람기에 의하여 자동적으로 선택리용될수 있는 메쏘드들은 다음과 같다.

init( )는 주클라스실체의 초기화작업을 완성하는데 리용한다.

start( )는 열람기를 시동하여 Applet의 주토막처리를 실행하는데 쓰인다.

paint( )는 Applet의 대면에서 문자, 도형, 기타 대면요소를 현시하는데 쓰인다.

stop( )는 start( )메쏘드의 반대조작이다.

destroy( )는 현재 Applet실체를 결속할 때 자원해방과 접속닫기 등 일부 조작을 완성한다.

**5.1.8. HTML파일이 전달하는 옹근수파라메터를 접수하고 그에 근거하여 본문칸의 길이를 지정하는 Applet프로그람을 작성하시오. 대응하는 HTML파일을 작성하여 이 Applet를 실행하시오.**

원천프로그람 ch5_e5_8.java

```
import java.applet.*;
import java.awt.*;
public class ch5_e5_8 extends Applet
{
   TextField tf = new TextField( );
   int tfLength = 0;
   public void init( )
   {
      try
      {
         tfLength = Integer.parseInt(getParameter("length"));
         tf.setColumns(tfLength);
         add(tf);
      }
      catch(NumberFormatException nfe)
      {
         tf.setText("HTML파일이 전달하는 파라메터형식이 틀립니다.");
         add(tf);
      }
   }
}
```

원천프로그람 ch5_e5_8.html

```
<html>
    <head>
       <title>ch5_e5_8</title>
    </head>
    <body>
       <hr>
          <applet code = ch5_e5_8 width = 700 height = 150>
             <param name = length value = 20>
          </applet>
       <hr>
    </body>
</html>
```

**5.1.9.** Applet의 paint( )메쏘드에 어떤 형의 형식파라메터가 있는가? 그것을 리용하여 어떤 조작을 완성할수 있는가?

paint( )메쏘드에는 하나의 형식파라메터가 있는데 그것이 Graphics이다. Graphics를 리용하여 도형, 문자, 기타 대면요소를 현시할수 있다.

**5.1.10.** 지령행파라메터가 지정하는 옹근수범위를 접수하여 이 범위안의 모든 완전수를 출력하는 Application프로그람을 작성하시오.

```
원천프로그람 ch5_e5_10.java
public class ch5_e5_10
{
   public static void main(String args[ ])
   {
      if(args.length < 2)
      {
         System.out.println("프로그람을 실행하려면 응당 두개의 옹근수형
            지령행파라메터를 제공해야 한다." + "실례로: \n java ch5_e5_10 1 1000");
         System.exit(0);
      }
      try
      {
         int floor, ceiling, temp, count = 0;
         floor = Integer.parseInt(args[0]);
         ceiling = Integer.parseInt(args[1]);
         if(floor > ceiling)
         {
            temp = floor;
            floor = ceiling;
            ceiling = temp;
         }
         for(int i = floor; i < ceiling; i++)
         {
            int y = 0;
            for(int j = 1; j < i; j++)
            {
               if(i % j == 0)
                  y += j;
```

```
            }
            if(y == i)
            {
               System.out.print(i + "\t");
               count++;
               if(count % 6 == 0)
                  System.out.println( );
            }
         }
      }
      catch(NumberFormatException nfe)
      {
         System.out.println(nfe.toString( ));
         System.out.println("지령행파라메터는 옹근수형으로 된다.
                        실례" + "\n java ch5_e5_10 1 1000");
      }
   }
}
```

**5.1.11. 재귀호출이란 무엇이며 어떤 기본요소가 있는가? 1차원배렬의 모든 원소의 적을 구하는 재귀프로그람을 작성하시오.**

재귀호출은 직접적으로 혹은 간접적으로 자기 자체를 호출하는것을 말한다. 재귀호출의 2가지 기본요소의 하나는 어떤 방식으로 문제를 간단히 하고 자기 자체를 호출하는것이고 다른 하나는 재귀를 중지하는 재귀머리부이다.

```
원천프로그람 ch5_e5_11.java
public class ch5_e5_11
{
   public static void main(String args[ ])
   {
      RecursionExample sample = new RecursionExample( );
      System.out.println("배렬원소:\n" + sample.toString( ));
      System.out.println("배렬원소의 적: " + sample.multipleArray( ));
   }
}
class RecursionExample
{
   int DataArray[ ]={1, 2, 3, 4, 5, 6, 7, 8, 9};
```

```
        long multipleArray( )
        {
           return multipleOneStep(DataArray,DataArray.length - 1);
        }
        long multipleOneStep(int[ ] array, int index)
        {
           if(index == 0)
              return array[index];
           else
              return multipleOneStep(array, index - 1) * array[index];
        }
        public String toString( )
        {
           String s=" ";
           for(int i = 0; i < DataArray.length; i++)
           {
              s=s+DataArray[i] + "\t";
           }
           return s;
        }
     }
```

**5.1.12. 정렬이란 무엇이며 정렬산법에는 몇가지가 있는가? 그것들이 어떤 우결함을 가지고있으며 각각 어떤 경우에 리용하면 좋은가?**

정렬은 자료순서렬의 매개 자료원소들을 그에 포함되여있는 어떤 키값에 따라 큰것으로부터 작은것으로 혹은 작은것으로부터 큰것으로 배렬하는 과정이다.

정렬의 목적은 대다수 자료순서렬의 원소에 대하여 탐색을 편리하게 하자는데 있다.

자주 리용하는 정렬방법에는 거품정렬(bubble), 선택정렬, 삽입정렬, 통정렬(bucket), 쾌속정렬(quick) 등이 있다.

거품정렬의 산법과 코드는 비교적 간단하며 정렬을 기다리는 자료원소개수가 그리 많지 않은 경우에 리용하면 좋다. 산법의 복잡도는 $n^3$이다.

선택정렬과 삽입정렬의 산법복잡도는 $n^2$이다. 계산량이 매우 작다. 그중에 선택정렬이 요구하는 비교조작은 많고 이동조작은 적다. 삽입정렬의 비교조작은 적지만 이동조작이 많다.

통정렬의 복잡도는 n이다. 계산량은 최대로 적지만 비교적 많은 기억공간을 차지한다.

괘속정렬의 복잡도는 n이다. 실제로 리용할 때에는 정렬하려는 원소자체의 특징에 근거하여 정렬한다.

### 5.1.13. 사용자가 입력한 몇개의 문자렬을 접수하고 영어자모순서로 정렬하여 출력하는 Applet프로그람을 작성하시오. 2개이상의 정렬산법을 리용하시오.

원천프로그람 ch5_e5_13.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch5_e5_13 extends Applet implements ActionListener
{
   final String SORT_METHOD_NAME[ ] = {"거품정렬", "선택정렬"};
   Label prompt = new Label("정렬하려는 문자렬을 입력하시오(최대로 10개):");
   TextField input = new TextField(5);
   Button sortBubbleBtn = new Button(SORT_METHOD_NAME[0]);
   Button sortSelectBtn = new Button(SORT_METHOD_NAME[1]);
   //정렬전 순서를 보관하는 배렬
   String[ ] OrigArray = new String[10]; //정렬을 기다리는 자료를 보관하는 배렬
   String[ ] DataArray = new String[10]; //이미 입력한 자료의 통계
   int DataInputed = 0; //정렬과정을 보관하는 2차원배렬
   String[ ][ ] SortPro = new String[11][10];
   public void init( ) //초기화
   {
      for(int i = 0; i < 10; i++)
      {
         DataArray[i] = " ";
         OrigArray[i] = " ";
         SortPro[10][i] = " ";
         for(int j = 0; j < 10; j++)
         SortPro[i][j] = " ";
      }
      add(prompt);
      add(input);
      add(sortBubbleBtn); //표식자, 본문마당, 단추를 Applet에 추가
      add(sortSelectBtn);
      input.setText(" ");
```

```
        input.addActionListener(this);
        sortBubbleBtn.addActionListener(this);
        sortSelectBtn.addActionListener(this);
    }
    public void paint(Graphics g) //정렬전 과정을 인쇄
    {
        for(int i = 0; i < SortPro.length; i++) //2차원배렬의 행수
        for(int j = 0; j < SortPro[i].length; j++) //2차원배렬의 첫번째행의 자료개수
        {
            g.drawString(SortPro[i][j], 10 + 80 * j, 40 + 20 * i);
        }
    }
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource( ) == input) //본문마당에 입력하고 넣기건(Enter)을 누르기
        { //자료기록
            DataArray[DataInputed] = input.getText( );
            OrigArray[DataInputed] = DataArray[DataInputed];
            DataInputed++;
            if(DataInputed < 10)
            {
                prompt.setText("이미 " + DataInputed +"개 입력, " + "문자렬 계속입력");
                input.setText(" "); //다음 자료의 입력을 준비
            }
            else //이미 10개의 자료를 입력
            {
                prompt.setText("이미 10개의 자료가 입력되였으므로 다시 입력할수 없다.");
                input.setVisible(false); //본문마당을 숨기기
            }
        }
        if(e.getSource( ) == sortBubbleBtn) //사용자가 단추누르기, 정렬과정을 기동
        {
            for(int i = 0; i < DataArray.length; i++) //정렬하지 않은 본래자료기록
            SortPro[0][i] = DataArray[i];
            BubbleSortProcedure( ); //거품정렬메쏘드를 리용
            repaint( );
            for(int i = 0; i < DataArray.length; i++)
```

```
        DataArray[i] = OrigArray[i]; //정렬전 무질서한 순서를 회복
    }
    if(e.getSource( ) == sortSelectBtn)
    {
        for(int i = 0; i < DataArray.length; i++) //정렬되지 않은 본래자료기록
            SortPro[0][i] = DataArray[i];
        SelectSortProcedure( ); //선택정렬메쏘드 리용
        repaint( );
        for(int i = 0; i < DataArray.length; i++)
        DataArray[i] = OrigArray[i]; //정렬전 무질서한 순서를 회복
    }
}
void BubbleSortProcedure( )//거품정렬메쏘드
{
    int pass, i, exchangeCnt;
    String temp = " ";
    for(pass = 0; pass < DataArray.length; pass++) //여러번 훑어보기
    {
        exchangeCnt = 0;// 둘씩 교환하는 회수를 기록
        for(i = 0; i < DataArray.length - pass - 1; i++) //한번 훑어보기과정
        { //비교범위를 매번 훑어보고 한개를 축소
            //한번의 둘씩 비교교환 과정
            if(DataArray[i].compareToIgnoreCase(DataArray[i + 1]) > 0)
            {
                temp = DataArray[i];
                DataArray[i] = DataArray[i+1];
                DataArray[i + 1] = temp;
                exchangeCnt++;
            }
        }
        for(i = 0; i < DataArray.length; i++)
            SortPro[pass + 1][i] = DataArray[i]; //훑어보기후의 자료배렬상태를 기록
        //만일 한번도 교환하지 않았다면 완전히 정렬이 되였으며 순환할 필요가 없다.
        if(exchangeCnt == 0)
            return;
    }
}
```

```
      void SelectSortProcedure( )//선택정렬메쏘드
      {
         int pass, i, k;
         String temp = " "; //여러번 선택하여 순서가 있는것은 증가배렬
         for(pass = 0; pass < DataArray.length - 1; pass++)
         { //과정을 한번 선택하여 순서가 없는것은 감소배렬
            for(i = pass, k = i; i < DataArray.length; i++)
            //나머지 정렬되지 않은 자료가운데서 최소를 선택
               if(DataArray[i].compareToIgnoreCase(DataArray[k]) < 0)
                  k = i;
            temp = DataArray[pass]; //나머지 자료의 제일 앞에 배렬
            DataArray[pass] = DataArray[k];
            DataArray[k] = temp;
            for(i = 0; i < DataArray.length; i++)
               SortPro[pass + 1][i] = DataArray[i]; //선택이후의 자료배렬상태를 기록
         }
      }
   }
```

**5.1.14. 문제 5.1.13을 통정렬을 리용하여 실현할수 있는가? 구체적으로 어떻게 조작해야 하는가? 문자렬의 매 문자를 영어자모순서의 크기를 표시하는 옹근수값으로 보시오.**

원천프로그람 ch5_e5_14.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch5_e5_14 extends Applet implements ActionListener
{
   final String SORT_METHOD_NAME[ ] = {"거품정렬", "통정렬"};
   Label prompt = new Label("정렬하려는 문자렬을 입력하시오(최대로 10개):");
   TextField input = new TextField(5);
   Button sortBubbleBtn = new Button(SORT_METHOD_NAME[0]);
   Button sortSelectBtn = new Button(SORT_METHOD_NAME[1]);
   String[ ] OrigArray = new String[10]; //정렬전 순서를 보관하는 배렬
   String[ ] DataArray = new String[10]; //정렬을 기다리는 자료를 보관하는 배렬
   int DataInputed = 0; //이미 입력한 자료의 통계
   String[ ][ ] SortPro = new String[11][10]; //정렬과정을 보관하는 2차원배렬
   public void init( ) //초기화
```

```
{
    for(int i = 0; i < 10; i++)
    {
        DataArray[i] = " ";
        OrigArray[i] = " ";
        SortPro[10][i] = " ";
        for(int j = 0; j < 10; j++)
            SortPro[i][j] = " ";
    }
    add(prompt);
    add(input);
    add(sortBubbleBtn); // 표식자, 본문마당, 단추를 Applet에 추가
    add(sortSelectBtn);
    input.setText(" ");
    input.addActionListener(this);
    sortBubbleBtn.addActionListener(this);
    sortSelectBtn.addActionListener(this);
}
public void paint(Graphics g)// 정렬전과정을 인쇄
{
    for(int i = 0; i < SortPro.length; i++) //2차원배렬의 행수
        for(int j = 0; j < SortPro[i].length; j++) //2차원배렬의 첫번째행의 자료개수
        {
            try{
                g.drawString(SortPro[i][j], 10 + 80 * j, 40 + 20 * i);
            }catch(NullPointerException npe)
            {
                System.out.println(i + "," + j);
            }
        }
}
public void actionPerformed(ActionEvent e)
{
    if(e.getSource( ) == input) // 본문마당에 자료를 입력하고 넣기건을 누르기
    { //자료기록
        DataArray[DataInputed] = input.getText( );
        OrigArray[DataInputed] = DataArray[DataInputed];
```

```
            DataInputed++;
            if(DataInputed < 10)
            {
                prompt.setText("이미 "+DataInputed+"개 입력,"+"문자렬을 계속 입력하시오.");
                input.setText(" "); //다음 자료의 입력준비
            }
            else //이미 10개의 자료를 입력
            {
                prompt.setText("이미 10개의 자료를 입력하였으므로 다시 입력할수 없다");
                input.setVisible(false); //다른 본문마당을 밀봉
            }
        }
        if(e.getSource( ) == sortBubbleBtn) //사용자가 단추누르기, 정렬과정이 진행
        {
            for(int i = 0; i < DataArray.length; i++) //정렬전의 본래자료를 기록
                SortPro[0][i] = DataArray[i];
            BubbleSortProcedure( ); //거품정렬메쏘드리용
            repaint( );
            for(int i = 0; i < DataArray.length; i++)
                DataArray[i] = OrigArray[i]; //정렬전의 무질서를 회복
        }
        if(e.getSource( ) == sortSelectBtn)
        {
            for(int i = 0; i < DataArray.length; i++) //정렬전의 본래자료를 기록
                SortPro[0][i] = DataArray[i];
            BucketSortProcedure( ); //통정렬메쏘드의 리용
            repaint( );
            for(int i = 0; i < DataArray.length; i++)
                DataArray[i] = OrigArray[i]; //정렬전의 무질서를 회복
        }
    }
    void BubbleSortProcedure( ) //거품정렬메쏘드
    {
        int pass, i, exchangeCnt;
        String temp = " ";
        for(pass = 0; pass < DataArray.length; pass++) //여러번 훑어보기
        {
```

```
        exchangeCnt = 0; //둘씩 교환하는 회수를 기록
        for(i = 0; i < DataArray.length - pass - 1; i++) //한번 훑어보기과정
        { //비교범위를 매번 훑어보고 한개를 축소
         //한번에 둘씩 비교, 교환과정
           if(DataArray[i].compareToIgnoreCase(DataArray[i + 1]) > 0)
           {
              temp = DataArray[i];
              DataArray[i] = DataArray[i + 1];
              DataArray[i + 1] = temp;
              exchangeCnt++;
           }
        }
        for(i = 0; i < DataArray.length; i++)
           SortPro[pass + 1][i] = DataArray[i]; //훑어보기후의 자료배렬상태를 기록
        //한번도 교환이 진행되지 않았으면 완전히 정렬이 되였으므로 다시 순환할 필요가 없다
        if(exchangeCnt == 0)
           return;
     }
  }
  void BucketSortProcedure( )//통정렬메쏘드
  {
     //일시적으로 문자범위를 128개의 ASCII코드내부로 제한한다.
     //마지막 한렬에 이 행의 자료개수를 보관
     String bucket[ ][ ] = new String[128][DataInputed+1];
     int pass = 0; //훑어보기회수를 계수
     for(int i = 0; i < 128; i++)
        for(int j = 0; j < DataInputed; j++)
           bucket[i][j] = " ";
     int strLength = 0, doo = 0;
     StringBuffer sb;
     //길이가 다른 문자렬은 길이가 같아지도록 한다.
     for(int i = 0; i < DataInputed; i++)
        strLength = Math.max(strLength, DataArray[i].length( ));
     for(int i = 0; i < DataInputed; i++)
        if(DataArray[i].length( ) < strLength)
        {
           sb = new StringBuffer(DataArray[i]);
```

```
            for(int j = 0; j < strLength - DataArray[i].length( ); j++)
               sb.append((char)doo);
            DataArray[i] = sb.toString( );
         }
      do
      {
         for(int i = 0; i < 128; i++)
         {
            bucket[i][DataInputed] = "0";
         }
         for(int i = 0; i < DataInputed; i++) //분산훑어보기
         {
            int ch;
            if(DataArray[i].length( ) - 1 >= pass)
               ch = DataArray[i].charAt(DataArray[i].length( ) - 1 - pass);
            else
               ch = 0;
            if(ch >= 128 || ch < 0)
            {
               showStatus("ASCII문자만을 처리할수 있다.");
               return;
            }
            if(ch >= 'A' && ch <= 'Z') //대소문자의 구별을 취소
            {
               ch += 'a' - 'A';
            }
            int count = Integer.parseInt(bucket[ch][DataInputed]);
            bucket[ch][count++] = DataArray[i];
            bucket[ch][DataInputed] = Integer.toString(count);
         }
         int k = 0;
         for(int i = 0; i < 128; i++) //집중 훑어보기
            for(int j = 0; j < Integer.parseInt(bucket[i][DataInputed]); j++)
               DataArray[k++] = bucket[i][j];
         for(int i = 0; i < DataArray.length; i++)//선택이후의 자료배렬상태를 기록
            SortPro[pass][i] = DataArray[i];
         pass++;
      }while(Integer.parseInt(bucket[0][DataInputed]) != DataInputed); //한번 훑어보기
   }
}
```

**5.1.15. 탐색이란 무엇이며 자주 리용하는 탐색산법에는 어떤것들이 있는가? 문제 5.1.13의 프로그람을 수정하여 사용자가 하나의 문자렬을 입력할때마다 그것을 영어자모순으로 적합한 위치에 보관하도록 하고 2분탐색법을 리용하여 적당한 삽입위치를 탐색하게 하시오.**

탐색은 주어진 한쌍의 키값을 리용하여 자료모임에서 혹은 자료순서렬에서 한쌍의 조건에 부합되는 하나의 자료를 추출해나가는 과정이다.

자주 리용하는 탐색산법에는 순서탐색, 2분탐색, 나무탐색 등이 있다.

원천프로그람 ch5_e5_15.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class ch5_e5_15 extends Applet implements ActionListener
{
   Label prompt = new Label("삽입하려는 문자렬을 입력:");
   TextField input = new TextField(5);
   Vector dataVector = new Vector( );
   public void init( )
   {
      add(prompt);
      add(input);
      dataVector.removeAllElements( );
      input.setText(" ");
      input.addActionListener(this);
   }
   public void paint(Graphics g) //인쇄
   {
      int i = 0;
      for(Enumeration e = dataVector.elements( ); e.hasMoreElements( ); i++)
      {
         try
         {
            g.drawString((String)(e.nextElement( )), 10 + 80 * i, 40);
         }
         catch(NullPointerException npe)
         {
```

```
                System.out.println(i);
            }
        }
    }
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource( ) == input) //본문마당에 입력하고 넣기건을 누르기
        { //자료기록
            String s = input.getText( );
            int low=0, high = dataVector.size( ) - 1, mid;
            while(low <= high)
            {
                mid = (high + low) / 2;
                if(((String)(dataVector.get(mid))).compareToIgnoreCase(s) == 0)
                {
                    dataVector.insertElementAt(new String(s), mid);
                    System.out.println(s + "," + mid);
                    break;
                }
                else if(((String)(dataVector.get(mid))).compareToIgnoreCase(s) > 0)
                    high = mid - 1;
                else
                    low = mid + 1;
            }
            if(low > high)
            {
                dataVector.insertElementAt(new String(s), low);
                System.out.println(s + "," + low);
            }
            input.setText(" ");
            repaint( );
        }
    }
}
```

**5.1.16. 목록이란 무엇이며 목록과 배렬의 다른 점은 무엇인가? 목록은 어떤 특수한 점이 있는가?**

목록은 자료마디점을 동적으로 생성, 기억, 삽입, 삭제, 이동할수 있는 유일한 선형자료구조이며 배렬은 길이가 고정된 자료원소를 련속적으로 블로크의 내부기억공간에 보관한다. 목록은 자료개수가 고정되지 않고 삽입, 삭제 등 변화가 비교적 많은것을 처리하는데 좋다.

**5.1.17. 대기렬이란 무엇이며 대기렬과 목록은 어떻게 다른가? 대기렬은 어떤 특징이 있는가? 대기렬자료구조를 리용하여 해결할수 있는 문제를 실례드시오.**

대기렬은 한끝은 입력으로 되고 다른 끝은 출력으로 되는 선형자료구조이다.

Java프로그람은 목록의 임의의 하나의 항목을 직접호출, 삽입, 삭제, 이동할수 있다. 즉 자료순서렬의 임의의 하나의 항목에 대하여 조작을 진행할수 있다. 그러나 대기렬은 선입선출의 원칙을 준수하므로 한개 항목을 호출하기전에 반드시 먼저 그 앞의 항목을 이동해야 한다. 즉 자료순서렬의 시작 혹은 마지막만을 호출할수 있다.

CPU의 다중과제분배대기렬, 인쇄기완충구역에서의 기다림작업대기렬, 망봉사기에서 처리를 기다리는 사용자요구대기렬 등은 모두 대기렬자료구조를 리용한 실례이다.

**5.1.18. 나무란 무엇이며 나무가 배렬, 목록, 대기렬, 탄창과 어떤 다른 점이 있는가? 마디점, 뿌리, 부분나무, 왼쪽부분나무의 개념을 말하시오. 2진나무와 2진탐색나무란 무엇인가?**

여기서 말하는 나무는 자연계의 나무와 비슷하다.

나무는 배렬, 목록, 대기렬, 탄창과 서로 다르다. 나무는 비선형자료구조이다.

마디점은 나무의 자료조직단위이며 매개 마디점은 한개의 자료와 뒤따르는 마디점들을 련결하는데 리용하는 몇개의 방향을 가진다. 뿌리는 앞에 련결된 마디점이 없는 마디점이다.

어떤 마디점의 부분나무는 그의 어떤 뒤따르는 마디점을 뿌리로 하는 나무이다.

왼쪽 부분나무는 현재 마디점의 왼쪽에 뒤따르는 마디점을 뿌리로 하는 모든 마디점들의 모임이다.

2진나무의 모든 마디점들은 최대로 두개의 뒤따르는 마디점을 가진다.

**5.1.19. 앞순서순회, 중간순서순회, 뒤순서순회란 무엇인가? 2진나무에서 그의 앞순서순회의 결과는 ABDGHECFI이고 중간순서순회의 결과는 GDHBEAFIC이다. 이 2진나무를 그리고 그의 뒤순서순회를 말하시오.**

2진나무를 순회할 때 중간마디점, 왼쪽부분나무, 오른쪽부분나무의 개수순서로 호출하는것을 앞순서순회라고 하고 왼쪽부분나무, 중간마디점, 오른쪽부분나무의 개수순서로 호출하는것을 중간순서순회라고 하며 왼쪽부분나무, 오른쪽부분나무, 중간마디점의 순서로 호출하는것을 뒤순서순회라고 한다.
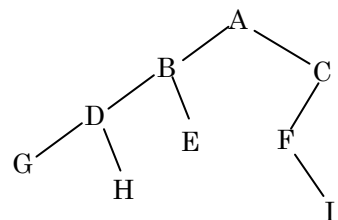
그림 5-1.

이 2진나무의 뒤순서순회는 GHDEBIFCA이다.

# 5.2. 보충문제

5.2.1. Java클라스의 조상클라스는 어느것인가? 거기서 매개 Java클라스가 공통으로 가지고있는 어떤 조작을 정의하는가?

5.2.2. Math클라스에 정의된 몇개의 수학계산기능을 어떻게 리용하는가? 왜 이 메쏘드가 모두 정적메쏘드로 정의되는가?

5.2.3. 1-100의 100개 옹근수를 우연적으로 만들어내는 Java프로그람을 작성하고 그것들의 평균값과 최대값, 최소값을 출력하시오.

5.2.4. Applet의 init( )메쏘드와 start( )메쏘드의 다른 점은 무엇이며 각각 어느 때에 실행되는가?

5.2.5. 2개의 단추 《확대》와 《축소》를 포함하고 사용자가 단추를 누를 때 Applet의 크기가 상응하게 확대, 축소되는 Applet를 작성하시오.

5.2.6. 배렬 dataArray[ ]가 있다고 할 때 그 길이를 구하는 명령을 쓰시오.

벡토르 myVector가 있다고 할 때 그 길이를 구하는 명령을 쓰시오.

문자렬 myString이 있다고 할 때 그의 문자개수를 구하는 명령을 쓰시오.

길이가 변하는 문자렬 myStringBuffer가 있다고 할 때 그의 문자개수를 구하는 명령을 쓰시오.

5.2.7. 아래의 배렬에서 틀린것을 찾고 고치시오.

1) int myArray = new int(15);

2) int myArray = new[15];

3) int myArray = new int{0, 1, 2, 3, 4};

4) int myArray = {0; 1; 2; 3; 4};

5) System.out.println(myArray.length( ));

6) System.out.println(myArray[myArray.length]).

5.2.8. 아래의 프로그람을 읽고 실행결과를 쓰시오.

```
public class test3
{
   public static void main(String args[ ])
   {
      int myArray[ ] = {0, 1, 2, 3, 4, 5};
      int i = 4;
      runMe(myArray,i);
      System.out.println("Array:");
      for(int j = 0; j < myArray.length; j++)
         System.out.print("\t" + myArray[j]);
```

```
        System.out.println("\nIndex:" + i);
    }
    public static void runMe(int arr[ ], int idx)
    {
        arr[idx]++;
        idx++;
    }
}
```

5.2.9. Vector클라스의 하위클라스를 작성하고 boolean isOrdered( )메쏘드를 추가하여 그 메쏘드가 Vector에 보관하고있는 원소들을 커지는 순서로 배렬할 때 true를 돌려주고 그렇지 않으면 false를 돌려주는 코드를 작성하시오.

5.2.10. 기본자료형인 1, -5.7과 문자렬형인 "1", "-5.7"사이에 호상 강제형변환하는데 필요한 메쏘드들을 총괄하는 표를 그리시오.

5.2.11. 아래의 프로그람을 고찰하고 틀린것을 고치시오.

```
public class test
{
    public static void main(String args[ ])
    {
        Vector myVector = new Vector( );
        myVector.add("Hello");
        System.out.println("첫 자모는" + myVector.get(0).charAt(0));
    }
}
```

5.2.12. 아래의 프로그람을 수정하여 현재 명령이 java sup_6_1 1 2 3 4 5일 때의 출구결과를 쓰고 콤퓨터로 실험하여 보시오. 실험에서 얻은 결과에 근거하여 이 프로그람을 어떻게 수정하면 보다 합리적인 출력을 얻을수 있는가를 고찰하시오.

```
public class sup_6_1
{
    public static void main(String args[ ])
    {
        if(args.length == 0)
        {
            System.out.println("————————————");
        }
        for(int i = 0; i < args.length; i++)
        {
            for(int j = 0; j < i; j++)
```

```
                System.out.print(args[i] + "\t");
            System.out.println( );
        }
    }
}
```

5.2.13. 문자렬 《irregular》에서 부분문자렬 《regular》를 취하는 명령을 쓰시오.

5.2.14. 아래의 명령을 읽고 그의 출력결과를 말해보시오.

```
String str = "lmnopqrst8253777";
System.out.println(str.indexOf(String.valueOf(str.indexOf("o"))));
System.out.println(str.substring(2, 7));
System.out.println(str.substring(9));
System.out.println(str.indexOf('r'));
System.out.println(str.indexOf('1'), str.indexOf('7'));
```

5.2.15. 아래의 문자렬을 영어자모순으로 배렬하시오.

《zip》, 《about》, 《At》, 《all》, 《0》

5.2.16. 아래의 프로그람을 읽고 그의 출력결과를 지적하시오. 왜 그렇게 되는가를 설명하시오.

```
public class test2
{
    public static void main(String args[ ])
    {
        String s1 = "Hello";
        String s2 = "Hello";
        String s3 = new String("Hello");
        System.out.println("s1" + (!s1.equals(s2)? "not": " ") + "equals s2");
        System.out.println("s1" + (!s1.equls(s3)? "not": " ") + "equals s3");
        System.out.println("s1" + (!(s1==s2)? "!=": "==") + "s2");
        System.out.println("s1" + (!(s1==s3)? "!=": "==") + "s3");
    }
}
```

5.2.17. 1개 문자렬에 있는 공백을 모두 없애는 메쏘드를 작성하고 처리후 얻어진 새로운 문자렬을 되돌리시오.

5.2.18. 사용자가 입력한 영어문자를 접수하고 배렬을 리용하여 자모총수에 대한 매개 자모가 나타나는 회수의 비률을 계산하여 현시 및 인쇄하는 프로그람을 작성하시오.

어느 자모가 나타나는 비률이 제일 높은가?

5.2.19. 형식파라메터가 문자렬이고 되돌이값이 론리형으로 되는 Java메쏘드를 작성하시오.

이 메쏘드는 문자렬이 합법적인 Java표식기호인가를 검사하고 만일 합법적인것이면 《참》을 되돌려주고 그렇지 않으면 《거짓》을 되돌리는 기능을 수행한다.

5.2.20. 사용자가 입력한 문자렬에 대하여 암호화를 하고 출력하는 암호화프로그람을 작성하시오. 암호화하는 메쏘드는 매개 문자를 그 자모표의 대칭문자로 한다.

즉 a는 z에 b는 y에 대응한다. 《Java》를 암호화하면 《Qzfz》로 된다.

5.2.21. 만일 2개의 문자렬에 포함된 문자가 완전히 같다면 이 두 문자렬을 철자가 바뀐 문자렬이라고 한다.(례: 《ACT》와 《CAT》) 사용자가 입력한 2개의 문자렬이 철자가 바뀐 문자렬인가를 검사하는 프로그람을 작성하시오.

5.2.22. 사용자가 입력한 몇개의 문자렬을 접수하고 보관하는 프로그람을 작성하시오. 문자렬을 입력할 때마다 그의 모든 철자가 바뀐 문자렬을 함께 인쇄하게 하시오.

5.2.23. 사용자가 입력한 지령행파라메터를 반대순서로 현시하는 Java Application 프로그람 text.java를 작성하시오. 실례로 입력명령행이 java text how are you doing이면 프로그람은 doing you are how를 출력한다.

5.2.24. 아래의 코드를 읽고 그 기능을 설명하시오.

```
1) public void myMethod1(int m)
   {
     if(m > 0)
     {
       System.out.println(m);
       MyMethod1(m - 1);
     }
     else
       return;
   }
2) public void myMethod2(String str)
   {
     if(str.length( ) > 0)
       System.out.print(str.charAt(0));
     else
     {
       System.out.print(str.charAt(str.length( ) - 1));
       MyMethod2(str.substring(0, str.length( ) - 1));
     }
   }
```

```
3) public void myMethod3(String str)
   {
      if(str.length > 0)
      {
         myMethod3(str.substring(1));
         System.out.print(str.charAt(0));
      }
   }
4) public String myMethod4(String str)
   {
      if(str.length( ) == 0)
         return " ";
      else if(str.charAt(0) == '/')
         return("\\" + myMethod4(str.substring(1)));
      else
         return (str.charAt(0) + myMethod4(str.substring(1)));
   }
```

5.2.25. 모든 재귀산법은 순환으로 대신하여 실현할수 있다. 이 말이 옳은가? 실례를 들어 설명하시오.

5.2.26. 아래의 메쏘드에서 순환을 재귀로 바꾸어 실현하시오.

```
int myMethod(int x)
{
   while(x > 0)
   {
      if(x % 2 == 1)
         x = (x - 1) / 2;
      else
         x = x / 2;
      System.out.println(x + "\t");
   }
   return x;
}
```

5.2.27. x의 y제곱을 구하는 재귀메쏘드를 작성하시오.

5.2.28. 최대원의 반경이 주어졌다. 제일 바깥층의 원을 제외하고 매 원의 반경은 다 바깥층의 반경에 비하여 10%작다. 원의 반경이 0으로 될 때까지 계렬동심원을 그리는 재귀메쏘드를 작성하시오.

5.2.29. 8bit 2진수자의 문자렬을 10진수자의 문자렬로 바꾸는 재귀코드를 작성하시오. 실례로 《00000111》을 《15》으로 바꾸시오.

5.2.30. 자료순서렬 8, 34, 9, 0, 217, 6, 32, 77, 95가 있다.

1) 콤퓨터의 정렬과정을 모방하여 아래의 매개 정렬방법에 대응하는 중간결과의 순서렬을 쓰시오.

① 거품정렬　　② 선택정렬　　③ 삽입정렬　　④ 통정렬

2) 콤퓨터의 탐색과정을 모방하여 아래의 탐색방법에서 비교탐색을 진행한 순서렬의 자료를 쓰시오.

① 순서탐색　　② 2분탐색

5.2.31. 2개의 순서가 있는 자료순서렬이 있다. 2개의 순서렬을 하나의 긴 순서렬로 합치는 프로그람을 작성하시오.

실현할 때 매개 본래순서렬에 대하여 단 한번만 훑어보기할것을 요구한다.

5.2.32. 탄창을 리용하여 하나의 문자렬이 회문인가 아닌가를 검사하시오.

5.2.33. 탄창을 리용하여 문자렬이 문법적규칙에 맞는가를 검사하시오. 검사규칙은 문자렬에서 쌍인용괄호, 소괄호, 중괄호, 대괄호들이 모두 쌍을 지어 나타날것을 요구한다.

5.2.34. Vector클라스로 대기렬자료구조를 실현하시오.

5.2.35. 도형대면의 대기렬을 실현하는 프로그람을 작성하시오. 거기에 《입구대기》, 《출구대기》라는 2개의 단추가 있어서 본문마당에 자료를 입력하고 즉시에 대기렬의 내용을 현시하는데 리용된다.

5.2.36. 목록에 대하여 아래와 같은 메쏘드를 만드시오.

1) insertAtEnd(Node): 형식파라메터를 목록의 마지막위치에 삽입한다.

2) size( ): 목록의 항목개수를 되돌려준다.

3) isEmpty( ): 목록이 비면 참을 되돌려주고 그렇지 않으면 거짓을 되돌려준다.

4) tail( ): 목록의 마지막항목을 되돌려준다.

5) append(Node): 파라메터목록을 현재목록의 마지막에 첨가하고 하나의 새로운 목록을 되돌려준다.

6) recursiveTraversal( ): 재귀방법을 리용하여 목록의 항목을 순회, 인쇄한다.

5.2.37. MyNode와 MyLinkedList클라스가 각각 항목과 목록을 표시한다고 하자. 여기서 head는 MyLinkedList의 첫번째 항목이고 isEmpty( )는 MyLinkedList의 메쏘드로서 목록이 비였는가를 검사하는데 리용된다.

MyMethod는 MyLinkedList의 다른 하나의 메쏘드이다. 아래의 코드를 읽고 기능을 설명하며 괄호안에 알맞는 문자렬을 써넣으시오.

```
public boolean myMethod(Object data)
{
  if(isEmpty( ))
  {
    System.out.println(________);  //(1)
    return false;
  }
  else
  {
    MyNode current = head;
    while(________ && !(current.getData( ).equals(data)))  //(2)
      current = current.getNext( );
    if(current.getData( ).equals(data))
    {
      System.out.println(_________);    //(3)
      return true;
    }
    else
    {
      System.out.println(_________);    //(4)
      ______________ ;                  //(5)
    }
  }
}
```

5.2.38. 목록클라스 LinkList의 하위클라스 OrderedLinkList를 정의하시오. 그의 원소들이 순서배렬되여있다. 만들어진 순서목록을 검사하는 프로그람을 작성하시오.

5.2.39. 1개 목록을 반대로 뒤집는 프로그람을 작성하시오.

# 제6장. 례외처리와 다중토막처리

## 6.1. 련습

### 6.1.1. 례외란 무엇이며 Java는 왜 례외처리기를 도입해야 하는가? 체계가 정의한 례외클라스는 례외처리기에서 어떤 작용을 하는가?

례외는 Java프로그람의 실행과정에 생기는 프로그람의 정상실행을 중지시키는 돌발적인 오유이다. Java의 례외처리기는 프로그람작성자가 프로그람을 실행할 때 생길수 있는 오유를 즉시에 효과적으로 처리한다.

체계가 정의한 례외클라스 Exception은 모든 례외클라스의 조상클라스이다.

이밖에도 자주 리용하는 실행례외를 정의한다. 실례로 입출력례외를 들수 있다.

### 6.1.2. 체계가 정의하는 3개의 실행례외를 드시오. 사용자프로그람에서 왜 례외를 자체로 정의해야 하는가? 사용자프로그람은 어떻게 례외를 정의하는가?

IOException, ArrayIndexOutOfBoundsException, NumberFormat Exception은 자주 리용되는 실행례외이다. 체계가 예견할수 있는 오유는 체계가 정의하는 실행례외로 처리할수 있지만 특유한 실행오유에 대해서는 사용자프로그람에서 사용자례외로서 자체로 정의해야 한다. 사용자는 프로그람에서 Exception의 하위클라스를 통하여 사용자례외를 정의할수 있다.

### 6.1.3. 체계례외와 사용자정의례외는 어떻게 던질수 있는가?

체계가 Java프로그람을 실행할 때 례외가 발견되면 프로그람에서 이 례외에 대한 처리조작을 정의하였는가를 먼저 검사한다. 만일 대응한 처리조작이 없으면 체계는 자동적으로 례외를 던질수 있으며 현재 프로그람실행을 중지시킨다. 사용자정의례외를 던지려면 throw명령을 리용하여야 한다.

### 6.1.4. 아래의 프로그람에서 틀린것을 찾으시오.

```
public class MyClass
{
   public static void main(String args[ ])
   {
      myMethod( );
   }
   public void myMethod( ) throw MyException
   {
      throws(new MyException( ));
      }
}
```

```
class MyException
{
   public String toString( )
   {
      return("사용자정의례외");
   }
}
```

이 프로그람은 다음과 같은것이 틀렸다.

(1) 사용자정의례외 MyException은 반드시 체계클라스 Exception의 파생클라스이여야 한다.

(2) 메쏘드를 정의하여 례외를 던지려면 예약어 throw가 아니라 throws를 리용하여야 한다.

(3) 례외를 던지는 명령은 예약어 throws가 아니라 throw이다.

(4) main( )메쏘드가 례외를 버릴수 있는 메쏘드 myMethod( )를 리용하기때문에 명령을 정의하여 myMethod( )가 main( )에게 던진 례외를 처리하여야 한다.

(5) main( )메쏘드는 정적인 static메쏘드이다. 그것은 오직 동일한 정적메쏘드 또는 정적마당만을 조작하고 선택리용할수 있으며 구체적인 객체에는 속하고 전체클라스에는 속하지 않는 myMethod( )메쏘드를 리용할수 없다. main( )메쏘드가 myMethod( )를 리용하기 위하여서는 반드시 myMathod( )를 static메쏘드로 정의하여야 한다.

수정한 프로그람:

```
public class MyClass
{
   public static void main(String args[ ])
   {
      try
      {
         myMethod( );
      }
      catch(Exception e)
      {
         System.out.println(e.toString( ));
      }
   }
   public static void myMethod( ) throws MyException
   {
      throw(new MyException( ));
   }
```

```
}
class MyException extends Exception
{
   public String toString( )
   {
      return("사용자 자체정의례외");
   }
}
```

**6.1.5. Java프로그람은 던져지는 례외를 어떻게 처리하며 어느것이 례외잡기를 진행하는가? 왜 try명령다음에 catch명령이 오는가? 매개 catch가 몇가지 례외를 처리할수 있으며 try에서 다중례외가 생길수 있다면 어떻게 처리해야 하는가?**

Java프로그람은 try명령을 통하여 례외처리기를 기동한다. try명령이 례외를 던지면 try뒤에 따르는 catch가 이 례외를 잡고 처리한다. 만일 던져진 례외형이 catch가 지적하는 례외형에 포함되여있지 않으면 메쏘드에서 이 례외를 처리할수 없으며 상층메쏘드로 귀환된다. 상층메쏘드에서도 이 례외를 처리할수 없으면 Java체계가 자체로 처리한다. 이때 프로그람의 집행을 중지할수 있으며 가상기계에서 탈퇴되여 조작체계로 돌아와 표준출력에 필요한 례외정보를 출구한다.

하나의 catch는 그 파라메터가 지적하는 례외 및 그 례외의 모든 하위클라스례외를 처리할수 있다. 만일 try에서 다중례외가 생길수 있다면 여러개의 catch명령을 정의하여 각각 처리하여야 한다.

**6.1.6. 프로그람, 처리와 토막처리(thread)사이의 관계를 간단히 서술하시오. 다중토막처리프로그람이란 무엇인가?**

프로그람은 정적상태의 코드이며 쏘프트웨어가 실행하는 원본이다.

처리는 프로그람의 한번의 동적실행과정이다. 토막처리는 처리에 비하여 보다 작은 프로그람실행단위이다. 하나의 처리는 다중토막처리를 리용하여 동시에 실행할수 있다. 서로 다른 토막처리사이에 동일한 기억구역과 자료를 공유할수 있다. 다중토막처리프로그람은 하나의 어떤 처리안에서 한개이상의 토막처리를 동시에 진행하는 프로그람이다.

**6.1.7. 토막처리에는 어떤 기본상태가 있으며 그것들사이에 어떻게 전환하는가? 토막처리의 생명주기를 간단히 설명하시오.**

토막처리에는 《창조》, 《준비완료》, 《실행》, 《막기》, 《소멸》의 5개 기본상태가 있다.

토막처리객체가 만들어질 때 《창조》상태에 들어간다. 프로그람실행명령은 토막처리가 대기렬에 들어가 CPU시간을 기다리게 한다. 이것이 《준비완료》상태이다.

준비완료상태의 토막처리가 처리기자원을 얻을 때 《실행》상태에 들어간다. 실행이 끝나거나 강제중지되면 《소멸》상태에 들어간다. 우에서 서술한 토막처리의 매개 상태들 사이의 전환은 토막처리의 기본생명주기를 이룬다.

**6.1.8. 토막처리도란 무엇인가? Java의 처리도는 어떤 원칙을 리용하는가?**

토막처리대기렬에서 CPU시간을 기다린 토막처리는 CPU에 의하여 봉사되는데 이것을 토막처리도라고 한다. Java의 처리도는 우선급에 기초하여 《먼저 도착한것이 먼저 봉사받는다》는 원칙에 따른다.

**6.1.9. Runnable대면에 어떤 추상메쏘드가 포함되는가? Thread클라스에 어떤 메쏘드가 있는가?**

Runnable대면에는 오직 하나의 추상메쏘드인 run( )메쏘드가 있다. Thread클라스의 중요한 메쏘드에는 run( ), start( ), sleep( ), isAlive( ) 등이 있다.

**6.1.10. Java프로그람에서 어떻게 다중처리를 실현하는가? Thread하위클라스를 리용하는것과 Runnable대면을 실현하는 2가지 방법의 차이점을 간단히 서술하시오.**

Java프로그람의 다중처리를 실현하는데서 가장 기본적인것은 부분토막처리조작을 정의하는것이다. 즉 run( )메쏘드를 정의하는것이다. 구체적으로 말하면 Thread클라스의 하위클라스를 파생하거나 Runnable대면을 실현하는 2가지 방법을 통하여 실현할수 있다.

Thread하위클라스를 파생할 때 상위클라스에서 내리적재한 run( )메쏘드를 통하여 부분토막처리의 구체적인 조작을 정의한 다음 주토막처리에서 그 하위클라스의 객체를 부분토막처리로 만들고 기동한다.

Runnable대면을 실현하는 클라스는 반드시 대면의 run( )메쏘드를 실현하여야 한다. 거기서 부분토막처리의 조작도 같이 정의한다. 그러나 이 메쏘드의 부분토막처리는 Thread하위클라스의 객체가 아니며 Thread클라스의 객체이다. Thread클라스객체를 만들 때 Runnable대면과 run( )메쏘드를 실현하는 클라스를 파라메터로 하여 그 객체에게 전달해주는것으로 한다. 이렇게 그의 구체적인 조작을 규정한다.

**6.1.11. 다중토막처리기술을 리용하여 Applet프로그람을 작성하시오. 1개의 문자렬이 왼쪽으로부터 오른쪽을 향하여 이동하며 모든 문자는 다 화면의 오른쪽에서 없어진후 새로 왼쪽에서 나타나고 계속 오른쪽을 향하여 이동한다.**

원천프로그람 ch6_e6_11.java

```
import java.awt.*;
import java.applet.*;
public class ch6_e6_11 extends Applet implements Runnable
{
   final String rollingMessage = "교육성 교육정보쎈터";
```

```
    Thread m_Draw = null;
    int beginX;
    public void init( )
    {
        m_Draw = new Thread(this);
    }
    public void paint(Graphics g)
    {
        g.drawString(rollingMessage, beginX, 40);
    }
    public void start( )
    {
        m_Draw.start( );
        try{
            Thread.sleep(50);
        }catch(InterruptedException e){ }
    }
    public void run( )
    {
        try
        {
            while(true)
            {
                beginX = ++beginX % getWidth( );
                repaint( );
                Thread.sleep(100);
            }
        }
        catch(InterruptedException e){ }
    }
}
```

# 6.2. 보충문제

6.2.1. catch를 정의하는 catch명령과 메쏘드를 정의하는 메쏘드머리부가 다른 점은 무엇인가?

6.2.2. 실행할 때 NullPointerException 혹은 ArrayIndexOutofBoundersException중의 하나를 우연적으로 던지는 간단한 Java프로그람을 작성하고 이 프로그람을 실행하여 례외를 어떻게 던지는가를 고찰하시오. 그리고 프로그람을 수정하여 실행시 던져진 례외를 프로그람자체로 처리하게 하시오.

6.2.3. 아래의 코드를 읽고 그의 출력을 말하시오.

```
public class ch6_e6_2_3
{
   public static void main(String args[ ])
   {
      int x = 0, y = -5;
      try
      {
         if(x >= 0)
            throw new ArithmeticException("Not a negative number.");
         if(y <= 0)
            throw new ArithmeticException("Not a positive number.");
      }
      catch(ArithmeticException ae)
      {
         System.out.println(ae.getMessage( ));
      }
   }
}
```

6.2.4. 아래의 프로그람을 읽고 틀린점을 지적하고 고치시오.

```
public static void main(String args[ ])
{
   try
   {
      String str = System.in.readLine( );
      if(Integer.parseInt(str) < 0)
         Throw new Exception("ERROR:I do not want a negative number!");
   }
```

```
        catch(NumberFormatException nfe)
        {
            System.out.println(nfe.getMessage( ));
        }
    }
```

6.2.5. 아래의 프로그람을 읽고 그 실행결과를 예측하시오. 이 프로그람을 실행하여 예측을 검사하시오. 이 프로그람은 어떻게 수정하여야 하는가?

```
public class test4
{
    public static void main(String args[ ])
    {
        for(int i = 1; i < 10; i++)
        for(int j = 1; j < i; j++)
            myMethod(i,j);
    }
    static void myMethod(int m, int n)
    {
        System.out.println(m + "/" + n + "=" + (m / n));
    }
}
```

6.2.6. 사용자가 입력한 2개의 자료를 원의 중심으로 하고 반경을 50으로 하는 원을 그리는 프로그람을 작성하시오. 례외처리기를 리용하여 사용자가 입력한 자리표가 부의 옹근수인 경우를 처리하시오.

6.2.7. 아래의 문장을 실행할 때 어떠한 례외를 발생시키는가?

(1) integer.parseInt("-1.81");

(2) int MyArray = new int[3];

System.out.println("The last number is: " + MyArray[3]);

(3) String str;

System.out.println("The first number is: " + str.charAt(0));

(4) String str = "Rolling";

System.out.println("The last leter is: " + str.charAt(str.length));

6.2.8. 아래의 례외들가운데서 검사코드를 추가하여 처리할수 있는것은 어느것인가? 처리할수 없는것은 어느것인가?

(1) ArrayIndexOutofBoundsException

(2) IOException

(3) NumberFormatException

(4) NullPointerException

(5) ClassNotFoundException

(6) ArithmeticException

6.2.9. 사용자가 입력한 이름, 전자우편주소, 전화번호를 접수하는 도형대면의 Java프로그람을 사용자정의례외클라스로 작성하시오.(이름과 전자우편주소는 비울수 없다.)

6.2.10. 우의 문제에 기초하여 2개의 본문마당을 추가하고 사용자가 입력한 이름과 암호를 접수하는 도형대면의 Java프로그람을 작성하는데 자체정의례외를 만들어 우에서 말한 요구를 실현하시오. 사용자이름의 길이는 8개문자를 초과할수 없으며 암호의 길이는 4개문자를 초과할수 없다. 그리고 빈칸을 포함할수 없다.

6.2.11. 토막처리의 실행을 어떻게 정지하는가? 한개의 토막처리가 다른 토막처리를 사멸시킬수 있는가? 또 자기 자체를 사멸시킬수 있는가?

6.2.12. Runable대면을 실현하는 클라스를 작성하시오. 이 클라스에 대응하는 토막처리가 기동된 후 1～1000사이에 있는 씨수를 인쇄하시오.

# 제7장. Java의 입출력

## 7.1. 련습

**7.1.1. Java의 입출력클라스서고란 무엇인가? Java의 기본입출력클라스는 무엇이며 흐름식입출력의 특징은 무엇인가?**

Java의 입출력클라스서고는 java.io패키지이다. Java의 기본입출력클라스는 InputStream클라스와 OutputStream클라스이다. 흐름식입출력의 특징은 자료의 쓰기, 읽기이다. 자료흐름순서렬의 순서에 따라 진행한다.

**7.1.2. 사용자가 입력한 10개의 옹근수형자료를 차례로 접수하여(매개 자료는 한개의 행이다.) 자료들을 커지는 순서로 정렬한후 체계의 표준출구로 출력하는 문자대면의 Application프로그람을 작성하시오.**

```
원천프로그람 ch7_e7_2.java
import java.io.*;
import java.util.*;
public class ch7_e7_2
{
   public static void main(String args[ ])
   {
      final int NUMBER = 10;
      Vector dataVector = new Vector( );
      try
      {
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         System.out.print("입력" + NUMBER + "개의 옹근수");
         //적합한 위치에 삽입
         for(int i = 0; i < NUMBER; i++)
         {
            int temp = Integer.parseInt(br.readLine( ));
            int low = 0, high = i - 1, mid = 0;
            while(low <= high)
            {
               System.out.println(low + "," + mid + "," + high);
               mid = (low + high) / 2;
               if(((Integer)dataVector.get(mid)).intValue( ) == temp)
```

```
                {
                    dataVector.insertElementAt(new Integer(temp), mid);
                    break;
                }
                else if(((Integer)dataVector.get(mid)).intValue( ) > temp)
                {
                    high = mid - 1;
                }
                else
                {
                    low = mid + 1;
                }
            }
            if(low > high)
            {
                dataVector.insertElementAt(new Integer(temp), low);
            }
        }
        //출력
        System.out.println("\n커지는 순서로 정렬한 결과는 :");
        for(int i = 0; i < NUMBER; i++)
        {
            System.out.print(dataVector.get(i).toString( ) + "\t");
        }
    }
    catch(NumberFormatException nfe)
    {
        System.out.println(nfe.toString( ));
        System.out.println("옹근수형식의 입력이 틀림");
    }
    catch(IOException ioe)
    {
        System.out.println(ioe.toString( ));
    }
  }
}
```

**7.1.3. Java프로그람은 어떤 클라스를 리용하여 파일을 관리하고 처리하는가? 1개의 명령을 작성하여 C구동기의 windows등록부에 myJavaPath부분등록부를 만드시오.**

Java프로그람은 File클라스를 리용하여 파일을 관리하고 처리한다. 아래의 명령은 C의 windows등록부에 부분등록부 myJavaPath를 만든다.

```
File myDir = new File("C:\windows\myJavaPath");
if(!myDir.exist( ))
   myDir.mkdir( );
```

**7.1.4. 사용자가 입력한 5개의 류동소수점수와 1개의 파일이름을 접수하여 5개의 자료를 그 파일에 보관하는 도형대면의 Java Application프로그람을 작성하시오.**

원천프로그람 ch7_e7_4.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class ch7_e7_4
{
   public static void main(String args[ ])
   {
      new GetDataStoreInFileFrame( );
   }
}
class GetDataStoreInFileFrame extends Frame implements ActionListener
{
   final int NUMBER = 5;
   final String DOUBLE_INPUT_PROMPT_PREFIX = "입력하시오.";
   final String DOUBLE_INPUT_PROMPT_SUFFIX = "번째 류동소수점수";
   final String FILE_NAME_INPUT_PROMPT = "자료를 보존하는 파일이름을 입력하시오.";
   double dataArray[ ] = new double[NUMBER];
   TextField inputTfd = new TextField(10);
   Label promptLbl = new Label(FILE_NAME_INPUT_PROMPT);
   int inputedData = 0;
   GetDataStoreInFileFrame( )
   {
      super("자료를 입력");
      promptLbl.setText(DOUBLE_INPUT_PROMPT_PREFIX
           + "1" + DOUBLE_INPUT_PROMPT_SUFFIX);
      inputTfd.addActionListener(this);
```

```
        setLayout(new FlowLayout( ));
        add(promptLbl);
        add(inputTfd);
        addWindowListener(new winClose( ));
        setSize(300, 200);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent ae)
    {
        try
        {
            if(ae.getSource( ) == inputTfd)
            {
                if(inputedData < NUMBER)
                {
                    dataArray[inputedData]=Double.parseDouble(inputTfd.getText( ));
                    inputTfd.setText(" ");
                    if(inputedData == NUMBER - 1)
                        promptLbl.setText(FILE_NAME_INPUT_PROMPT);
                    else
                        promptLbl.setText(DOUBLE_INPUT_PROMPT_PREFIX
                            + (inputedData+2)+DOUBLE_INPUT_PROMPT_SUFFIX);
                    inputedData++;
                }
                else
                {
                    File dataFile = new File(inputTfd.getText( ).trim( ));
                    FileWriter fw = new FileWriter(dataFile);
                    for(int i = 0; i < NUMBER; i++)
                    {
                        String str = Double.toString(dataArray[i]);
                        fw.write(str, 0, str.length( ));
                        fw.write('\n');
                    }
                    fw.close( );
                    inputTfd.setEnabled(false);
                    promptLbl.setText("자료는 이미 파일에 보관되여있다.");
```

```
                }
            }
        }
        catch(NumberFormatException nfe){ }
        catch(IOException ioe){ }
    }
}
class winClose extends WindowAdapter
{
    public void windowClosing(WindowEvent we)
    {
        we.getWindow( ).dispose( );
        System.exit(0);
    }
}
```

주의: 이 프로그람은 류동소수점수를 문자렬로 변환하여 보관한다. 만일 류동소수점수를 그대로 보관하려면 RandomAccessFile클라스를 리용해야 한다.

**7.1.5. 문제 7.1.4의 프로그람을 수정하고 FileDialog클라스를 리용하여 파일이름을 선택하시오.**

```
원천프로그람 ch7_e7_5.java
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class ch7_e7_5
{
    public static void main(String args[ ])
    {
        new GetDataStoreInFileFrame( );
    }
}
class GetDataStoreInFileFrame extends Frame implements ActionListener
{
    final int NUMBER = 5;
    final String DOUBLE_INPUT_PROMPT_PREFIX = "입력하시오.";
    final String DOUBLE_INPUT_PROMPT_SUFFIX = "번째 류동소수점수";
    final String FILE_NAME_INPUT_PROMPT = "자료를 보존하는 파일이름을 입력하시오.";
    double dataArray[ ] = new double[NUMBER];
```

```
TextField inputTfd = new TextField(10);
Label promptLbl = new Label(FILE_NAME_INPUT_PROMPT);
int inputedData = 0;
GetDataStoreInFileFrame( )
{
   super("자료를 입력");
   promptLbl.setText(DOUBLE_INPUT_PROMPT_PREFIX
      + "1" + DOUBLE_INPUT_PROMPT_SUFFIX);
   inputTfd.addActionListener(this);
   setLayout(new FlowLayout( ));
   add(promptLbl);
   add(inputTfd);
   addWindowListener(new winClose( ));
   setSize(300, 200);
   setVisible(true);
}
public void actionPerformed(ActionEvent ae)
{
   try
   {
      if(ae.getSource( ) == inputTfd)
      {
         if(inputedData < NUMBER)
         {
            dataArray[inputedData] = Double.parseDouble(inputTfd.getText( ));
            inputTfd.setText(" ");
            if(inputedData != NUMBER-1)
               promptLbl.setText(DOUBLE_INPUT_PROMPT_PREFIX
                  +(inputedData+2)+DOUBLE_INPUT_PROMPT_SUFFIX);
            else
            {
               promptLbl.setText(FILE_NAME_INPUT_PROMPT);
               inputTfd.setVisible(false);
               FileDialog fdialog = new FileDialog(this, "파일열기", FileDialog.LOAD);
               fdialog.setDirectory("c:\\temp");
               fdialog.show( );
               File dataFile = new File(fdialog.getDirectory( ),
               fdialog.getFile( ));
               RandomAccessFile raf = new RandomAccessFile(dataFile, "rw");
```

```
                    for(int i = 0; i < NUMBER; i++)
                    {
                       raf.writeDouble(dataArray[i]);
                    }
                    //검증
                    raf.seek(0);
                    while(raf.getFilePointer( ) < raf.length( ))
                    {
                       System.out.println(raf.readDouble( ) + "\t");
                    }
                    raf.close( );
                    inputTfd.setEnabled(false);
                    promptLbl.setText("자료는 이미 파일에 보관되여있다.");
                 }
                 inputedData++;
              }
           }
        }
        catch(NumberFormatException nfe){ }
        catch(IOException ioe){ }
     }
  }
  class winClose extends WindowAdapter
  {
     public void windowClosing(WindowEvent we)
     {
        we.getWindow( ).dispose( );
        System.exit(0);
     }
  }
```

**7.1.6. RandomAccessFile클라스가 다른 입출력클라스와 다른점은 무엇이며 그것은 어떤 대면을 실현하는가? 어떤 비교적 강한 입출력기능을 가지고있는가?**

RandomAccessFile클라스가 다른 파일입출력클라스와 다른점은 우선 파일의 임의의 위치에서 우연적으로 읽기를 실현할수 있으며 다음으로 임의의 형의 자료읽기를 실현할수 있다는것이다.

그것은 DataInput와 DataOutput의 2개의 대면을 실현한다.

# 7.2. 보충문제

7.2.1. InputStream클라스와 그 하위클라스가 함께 가지고있는 메쏘드를 모두 쓰시오. InputStream과 DataInputStream이 어떻게 다른가?

7.2.2. 아래의 코드를 읽고 그 기능을 말하시오.

(1)
```
PrintWriter pw = new PrintWriter(new FileWriter("MyFile.txt"));
pw.print("This is my file.");
pw.close( );
```

(2)
```
BufferedReader br = new BufferedReader(new FileReader("MyFile.txt"));
String str=br.readLine( );
while(str != null)
{
   System.out.println(str);
   str = br.readLine( );
}
BufferedReader br = new BufferedReader(new FileReader("MyFile.txt"));
int ch = br.read( );
while(ch != -1)
{
   System.out.print((char)ch);
   ch = br.read( );
}
```

(3)
```
DataOutputStream dout = new DataOutputStream(new FileOutputStream("MyBinaryFile"));
for(int j = 0; j < 7; j++)
{
   dout.writeUTF("주" + (j + 1));
   dout.writeDouble(30 + j); //최고온도
   dout.writeDouble(20 + j); //최저온도
   dout.writeBoolean(j % 2 == 1? true: false); //비가 오는가
}
dout.close( );
```

(4)
```
DataInputStream din = new DataInputStream(new FileInputStream("MyBinaryFile"));
try
{
   for(;true;)
   {
```

```
            System.out.print(din.readUTF( ) + "\t");
            System.out.print("최고온도" + din.readDouble( ) + "\t");
            System.out.print("최저온도" + din.readDouble( ) + "\t");
            System.out.print(((din.readBoolean( ))? "있다": "없다") + "비");
        }
    }
    catch(EOFException eofe) { }
    finally{
        din.close( );
    }
```

7.2.3. 2진파일의 읽기와 문서파일의 읽기의 다른 점이 무엇이며 각각 어떤 흐름클라스를 리용해야 하는가? 만일 파일이 문서파일인지 2진파일인지 미리 알수 없다면 공통메쏘드로 파일읽기를 실현할수 있는가?

7.2.4. 기후클라스를 실현하는 프로그람을 작성하시오. 매개 객체에는 월, 날자 그때의 평균온도와 비오는 상태가 포함된다. 동시에 ObjectInputStream, ObjectOutputStream, FileInputStream, FileOutputStream으로 기후객체전체를 파일에 읽어들이고 불러내는 두가지 메쏘드를 실현하시오. 프로그람을 실행하여 코드를 검사하시오.

7.2.5. 아래와 같은 지령행파라메터를 리용하여 2개의 파일이름을 지적하고 하나의 파일을 다른 파일로 복사하는 프로그람을 작성하시오.

java Mycopy 원천파일 목표파일

7.2.6. 2개의 파일이 있다. 매개 파일에 커지는 순서로 배렬된 수자가 있다. 이 2개의 파일의 수자를 합하고 커지는 순서로 3번째 파일에 보관하시오.

7.2.7. 문서파일을 여는 Java메쏘드를 작성하시오. 거기서 지적된 단어가 나타나는 회수를 계수하시오.

7.2.8. FileDialog를 리용하여 문서파일을 열고 파일내용을 문서구역에 입력한 다음 문서파일의 문자수, 단어수, 문장수를 계수하는 도형대면의 Java프로그람을 작성하시오.

7.2.9. 사용자가 입력한 하나의 목록이름을 접수하여 이 목록안의 Java원천프로그람 파일개수, 파일이름, 바이트코드 파일개수, 파일이름을 계수하고 출력하는 Java 프로그람을 작성하시오.

# 제8장. 도형사용자대면부의 설계와 실현

## 8.1. 련습

### 8.1.1. 도형사용자대면부란 무엇이며 그것이 문자대면과 어떤 다른 점이 있는가? 도형사용자대면부에서 리용하는 부분품을 말하시오.

도형사용자대면부는 도형방식의 리용 즉 차림표와 단추 등 표준대면요소와 마우스조작의 도움을 받아 사용자가 편리하게 콤퓨터체계에 명령을 주고 조작을 진행하며 체계실행의 결과를 도형방식으로 현시하는것을 말한다.

문자대면은 단순히 문자를 콤퓨터체계의 입출력으로 리용한다. 문자명령에 숙련된 사용자는 복잡한 조작을 능히 할수 있지만 문자가 아닌 정보를 처리할수 없다. 문자대면은 도형대면과 같이 편리하고 직관적이지 못하다. 도형대면에서 자주 리용하는 부분품에는 차림표, 본문마당, 표식자, 선택단추, 목록, 단추, 흘림띠 등이 있다.

### 8.1.2. 도형대면의 구성성분과 그의 작용을 말하시오. 그리고 도형사용자대면부를 설계하고 실현하자면 어떻게 하여야 하는가?

Java에서 도형대면의 구성성분은 3가지로 분류할수 있다. 즉 용기, 조종부분품, 사용자정의성분이다. 용기는 다른 대면성분을 조직하고 포함하는데 리용된다. 조종부분품은 사용자와의 직접적인 접촉을 실현하는 최소단위이다. 사용자정의성분은 항상 현시기능만을 가지며 사용자의 입력을 접수할수 없다. 도형사용자대면부를 설계하고 실현하자면 매개 성분을 만들고 조직한 다음 매개 성분의 기능을 정의하여야 한다.

### 8.1.3. Java프로그람의 도형사용자대면부에는 어떤 사용자정의성분들이 있는가?

기하도형, 문자, 색조절, 화상, 동화상 등은 모두 사용자가 Java도형사용자대면부에서 자체로 정의할수 있는 성분들이다.

### 8.1.4. 나사가 회전하는것을 선으로 그리는 Applet프로그람을 작성하시오.

원천프로그람 ch8_e8_4.java

```
import java.awt.*;
import java.applet.Applet;
public class ch8_e8_4 extends Applet
{
   int width = 50, height = 50;
   int x = 100, y = 70;
   int startA = 0;
   public void paint(Graphics g)
```

```
        {
            for(int i = 0; i < 10; i++)
            {
                g.drawArc(x, y, width, height, startA, 90);
                if(i % 2 == 0)
                {
                    width += 20;
                    x -= 10;
                }
                else
                {
                    height += 20;
                    y -= 10;
                }
                startA = (startA + 90) % 360;
            }
        }
    }
```

**8.1.5.** paint( )메쏘드를 리용하여 한개 문자렬을 현시하는 Applet프로그람을 작성하시오. Applet는 2개의 단추 《확대》, 《축소》를 포함하며 《확대》를 누를 때 문자렬크기는 확대되고 《축소》를 누를 때 문자렬크기는 축소된다.

원천프로그람 ch8_e8_5.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_5 extends Applet implements ActionListener
{
    String msgString = "교육성 교육정보쎈터";
    Button enlargeBtn = new Button("확대");
    Button dwindleBtn = new Button("축소");
    int currentFontSize = 12;
    public void init( )
    {
        add(enlargeBtn);
        add(dwindleBtn);
```

```
        currentFontSize = 12;
        enlargeBtn.addActionListener(this);
        dwindleBtn.addActionListener(this);
    }
    public void paint(Graphics g)
    {
        Font newFont, oldFont;
        oldFont = g.getFont( );
        newFont = new Font(oldFont.getFontName( ),
        oldFont.getStyle( ), currentFontSize);
        g.setFont(newFont);
        g.drawString(msgString, 10, 100);
    }
    public void actionPerformed(ActionEvent ae)
    {
        if(ae.getSource( ) == enlargeBtn)
            currentFontSize++;
        else if(ae.getSource( ) == dwindleBtn)
            currentFontSize-- ;
        System.out.println(currentFontSize);
        repaint( );
    }
}
```

**8.1.6. 3개의 표식자(label)를 포함하며 그 배경이 각각 붉은색, 노란색, 푸른색으로 되는 Applet프로그람을 작성하시오.**

```
원천프로그람 ch8_e8_6.java
import java.applet.*;
import java.awt.*;
public class ch8_e8_6 extends Applet
{
    Label redLbl = new Label("붉은색");
    Label yellowLbl = new Label("노란색");
    Label blueLbl = new Label("푸른색");
    public void init( )
```

```
        {
            redLbl.setBackground(Color.red);
            yellowLbl.setBackground(Color.yellow);
            blueLbl.setBackground(Color.blue);
            add(redLbl);
            add(yellowLbl);
            add(blueLbl);
        }
    }
```

**8.1.7. 사용자가 .gif화상파일이름을 입력하도록 하고 이 화상파일을 기억기에 적재하고 현시하는 프로그람을 작성하시오.**

```
원천프로그람 ch8_e8_7.java
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_7 extends Applet implements ActionListener
{
    Label promptLbl = new Label("현시하려는 화상파일이름입력:");
    TextField inputTfd = new TextField(20);
    Button getImageBtn = new Button("화상현시");
    Image myImage;
    public void init( )
    {
        add(promptLbl);
        add(inputTfd);
        add(getImageBtn);
        inputTfd.setText(" ");
        getImageBtn.addActionListener(this);
    }
    public void paint(Graphics g)
    {
        if(myImage != null)
            g.drawImage(myImage, 10, 100, this);
    }
```

```
    public void actionPerformed(ActionEvent ae)
    {
        if(ae.getSource( ) == getImageBtn)
        {
            String str = inputTfd.getText( ).trim( );
            if(!(str.substring(Math.max(0, str.length( ) - 4)).equals(".gif")))
                str=str.trim( ) + ".gif";
            myImage = getImage(getDocumentBase( ), str);
            repaint( );
        }
    }
}
```

**8.1.8.** Applet에 2개의 단추 《왼쪽회전》, 《오른쪽회전》을 추가하고 사용자가 이 단추를 누르면 동화상이 대응하는 방향으로 회전하게 하시오.

원천프로그람 ch8_e8_8.java

```
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
public class ch8_e8_8 extends Applet implements ActionListener
{
    Image[ ] m_Images;
    int totalImages = 18;
    int currentImage = 0;
    int frameChange = 0;
    Button leftRotateBtn = new Button("왼쪽회전");
    Button rightRotateBtn = new Button("오른쪽회전");
    public void init( )
    {
        m_Images = new Image[totalImages];
        for(int i = 0; i < totalImages; i++)
            m_Images[i] = getImage(getDocumentBase( ), "images\\Img00" + (i + 1) + ".gif");
        add(leftRotateBtn);
        add(rightRotateBtn);
        leftRotateBtn.addActionListener(this);
```

```
        rightRotateBtn.addActionListener(this);
    }
    public void start( )
    {
        currentImage = 0;
    }
    public void paint(Graphics g)
    {
        g.drawImage(m_Images[currentImage], 50, 50, this);
        currentImage = currentImage + frameChange;
        if(currentImage < 0)
            currentImage += totalImages;
        else
            currentImage %= totalImages;
        try{
            Thread.sleep(50);
        }
        catch(InterruptedException e)
        {
            showStatus(e.toString( ));
        }
        repaint( );
    }
    public void actionPerformed(ActionEvent ae)
    {
        if(ae.getSource( ) == leftRotateBtn)
            frameChange = -1;
        else if(ae.getSource( ) == rightRotateBtn)
            frameChange = 1;
    }
}
```

**8.1.9. Java의 사건처리기와 위탁사건모형을 간단히 말하시오. 사건원천이란 무엇이며 감시자란 무엇인가? Java의 도형사용자대면부에서 누가 사건원천을 충당하며 누가 감시자를 충당하는가?**

사건처리는 프로그람흐름과정에 주동적으로 진행되는것이 아니라 사용자에 의하여 프로그람이 리용될 때 즉흥적으로 진행된다.

이에 맞춰 사건처리기는 반드시 감시하는 기능을 가지고 아무때나 사용자의 동작을 감시하게 해야 한다. 감시기능을 실현하려면 반드시 사전에 정의된 사건이 있어야 하며 서로 다른 사건들이 어떤 사건원천들에 의하여 만들어지는가, 어떤 감시자에 의하여 처리되는가를 규정하여야 한다.

사건원천은 사건을 만들수 있는 도형사용자대면부의 조종부분품이다. 감시자에는 사건원천이 만들수 있는 사건을 처리하는 조작을 정의한다. 사건원천을 사건형에 따라 선정된 감시자에게 등록한후 체계는 사건원천이 만든 사건을 감시하고 감시자에서 정의한 조작을 자동적으로 리용하여 사건을 완성한다. 이것이 바로 Java의 사건처리기이다.

사건원천을 감시자에게 등록하고 감시자로 하여금 사건을 처리하게 하는것을 위탁사건모형이라고 부른다. 사건을 능히 만들수 있는것이 사건원천이며 사건을 처리하는 대면을 실현하는것이 감시자이다. Java도형사용자대면부에서 사건원천은 여러가지 조종부분품이며 감시자는 여러가지 용기부분품이다.

**8.1.10. java.awt.event패키지에서 정의한 사건클라스와 그것들의 계승관계를 말하시오.**

java.awt.event에는 계승관계를 가지고있는 사건클라스들이 정의되여있다.

사건클라스나무의 뿌리마디점은 EventObject클라스이며 그것은 AWTEvent클라스를 파생해낸다. AWTEvent클라스는 ActionEvent클라스, AdjustmentEvent클라스, ComponentEvent클라스, ItemEvent클라스, TextEvent클라스들을 파생해낸다.

그중에 ComponentEvent클라스는 다시 ContaintEvent클라스, FocueEvent클라스, InputEvent클라스, PaintEvent클라스, WindowEvent클라스들을 파생해낸다. 그중에 InputEvent클라스는 또 MouseEvent클라스, KeyEvent클라스들을 파생해낸다.

**8.1.11. GUI의 표준부분품을 리용하는 기본단계를 말해보시오.**

표준조종부분품을 리용하는 기본단계는 먼저 이 조종부분품을 만들고 그의 속성을 정의한 다음 그것을 어떤 용기의 적합한 위치에 배치한다. 마지막에 부분품의 감시자를 만들고 그것을 감시자에게 등록한다.

**8.1.12. 자주 사용하는 부분품의 창조명령, 메쏘드, 일어날수 있는 사건, 등록을 요구하는 감시자와 감시자가 내리적재하려는 메쏘드들을 하나의 표에 종합하여 서술하시오.**

부분품 및 창조명령, 자주 사용하는 메쏘드를 표 7-1에 주었다. 조종부분품이 일으킬 수 있는 사건, 감시자 및 그의 메쏘드는 표 7-2에 주었다.

표 7-1. 패키지 및 명령만들기, 자주 사용하는 방법

| 부분품 | 부분품창조명령 | 자주 사용하는 메쏘드 |
|---|---|---|
| Button | new Button( )<br>new Button(String) | getLabel( ), setLabel(String)<br>getActionCommand( ), setActionCommand( ) |
| Checkbox | new Checkbox( )<br>new Checkbox(String)<br>new Checkbox(String, boolean)<br>new Checkbox(String, boolean, CheckboxGroup) | getLabel( ), setLabel(String),<br>getState( ), setState(boolean),<br>getSelectedObjects( ),<br>setCheckboxGroup(CheckboxGroup) |
| Checkbox Group | new CheckboxGroup( ) | setSelectedCheckbox(Checkbox) |
| Label | new Label( )<br>new Label(String) | getText( ),<br>setText(String) |
| List | new List( )<br>new List(int visibleRowNumber) | add(String), add(String, int),<br>getItem(int), getItemCount( ),<br>getItems( ), getSelectedIndex( ),<br>getSelectedItems( ) |
| Menu | new Menu( )<br>new Menu(String)<br>new Menu(String, boolean) | add(MenuItem), add(String),<br>getItem(int),<br>insertItem(MenuItem, int), remove(int) |
| MenuItem | new MenuItem( ),<br>new MenuItem(String)<br>new MenuItem(String, MenuShortcut) | getActionCommand( ), getLabel( ),<br>setLabel(String),<br>setEnabled(boolean), isEnabled( ),<br>setActionCommand(String) |
| Scrollbar | new Scrollbar( )<br>new Scrollbar(int orientation, int value, int visible, int min, int max) | getValue( )<br>setValue( ) |
| TextArea | new TextArea( ),<br>new TextArea(int rows, int columns) | append(String), insert(String, int),<br>replaceRange(String, int start, int end),<br>getText( ), setText(String),<br>getSelectedText( ) |
| TextField | new TextField( ),<br>new TextField(int)<br>new TextField(String) | setEchoChar(char),<br>getSelectedText( ),<br>getText( ), setText(String) |

표 7-2. 일어날수 있는 사건, 감시자 및 그의 메쏘드

| 사건 | 사건원천 및 감시자를 등록하는 방 법 | 감시자 및 그의 사건처리메쏘드 | 사건클라스가 자주 리용하는 메쏘드 |
|---|---|---|---|
| ActionEvent | Button, TextField, List, MenuItem, addActionListener( ActionListener) | ActionListener actionPerformed( ActionEvent) | getSource( ), getActionCommand( ) |
| Adjustment Event | Scrollbar addAdjustmentEventListener( ) | AdjustmentListener adjustmentValueChanged( AdjustmentEvent) | getAdjustable( ), getValue( ) |
| ItemEvent | Checkbox, Choice, List, CheckboxMenuItem, addItemListener( ItemListener) | ItemListener itemStatChanged(ItemEvent) | getItem( ), getItemSelectable( ), getStateChange( ) |
| TextEvent | TextArea, TextField addTextListener( TextListener) | TextListener textValueChanged( TextEvent) | getSource( ) |
| MouseEvent | Canvas, Applet, Dialog, Frame, Panel addMouseListener( MouseListener), addMouseMotionListener( MouseMotionEvent) | MouseListener mouseClicked(MouseEvent), mouseEntered(MouseEvent), mouseExited(MouseEvent), mousePressed(MouseEvent), mouseReleased(MouseEvent), MouseMotionListener mouseDragged(MouseEvent), mouseMoved(MouseEvent) | getClickCount( ) getPoint( ), getX( ), getY( ), getSource( ) |
| KeyEvent | Canvas, Applet, Dialog, Frame, Panel addKeyListener( KeyListener) | KeyListener keyPressed(KeyEvent), keyReleased(KeyEvent), keyTyped(KeyEvent) | getKeyChar( ) getSource( ) |
| Window Event | Dialog, Frame | WindowListener windowActivated( WindowEvent), windowClosed( WindowEvent), windowClosing( WindowEvent), windowDeactivated( WindowEvent), windowIconified( WindowEvent), windowOpened( WindowEvent) | getWindow( ) |

### 8.1.13. 동작사건의 사건원천에는 어떤것들이 있는가? 어떻게 동작사건에 응답하는가?

단추, 본문부분품, 목록, 차림표선택항목은 모두 동작사건원천이다. 동작사건에 응답하려면 동작사건원천을 등록하고 actionPerformed( )메쏘드를 실현하여야 한다.

### 8.1.14. 1개의 표식자, 본문마당, 단추를 포함하며 사용자가 단추를 누를 때 프로그람이 본문마당의 내용을 표식자에 복사하는 Applet를 작성하시오.

원천프로그람 ch8_e8_14.java

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_14 extends Applet implements ActionListener
{
   Label outputLbl = new Label(" ");
   TextField inputTfd = new TextField(10);
   Button copyBtn = new Button("복사");
   public void init( )
   {
      inputTfd.setText(" ");
      outputLbl.setText(" ");
      add(inputTfd);
      add(outputLbl);
      add(copyBtn);
      copyBtn.addActionListener(this);
   }
   public void actionPerformed(ActionEvent ae)
   {
      if(ae.getSource( ) == copyBtn)
      {
         outputLbl.setText(inputTfd.getText( ));
      }
   }
}
```

**8.1.15. 본문마당과 본문구역의 창조방법, 자주 리용하는 메쏘드, 사건에 대한 응답에서 서로 다른 점이 있는가? 어떤 조작이 본문사건을 일으키는가? 본문사건에 어떻게 응답하는가? 1개의 본문마당, 본문구역과 단추를 포함하며 사용자가 단추를 누를 때 프로그람이 본문구역에서 선택된 문자렬을 본문마당에 복사하는 Applet를 작성하시오.**

본문마당(TextField)과 본문구역(TextArea)은 다 본문부분품이다. 그것들의 메쏘드와 사건에 대한 응답은 원리상 기본적으로 같다. 다른 점은 본문마당은 오직 1개 행의 본문만을 가지며 본문구역은 몇개 행의 본문을 가질수 있다는것이다. 또한 본문구역은 동작사건에 응답할수 없다.

본문부분품에 문자를 입력하거나 편집하는것은 본문사건을 일으킨다. 본문사건에 응답하려면 textValueChanged( )메쏘드를 실현하여야 한다.

원천프로그람 ch8_e8_15.java

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_15 extends Applet implements ActionListener
{
   TextField outputTfd = new TextField(30);
   TextArea inputTar = new TextArea(10,45);
   Button copyBtn = new Button("복사");
   public void init( )
   {
      add(inputTar);
      add(copyBtn);
      add(outputTfd);
      inputTar.setText(" ");
      outputTfd.setText(" ");
      copyBtn.addActionListener(this);
   }
   public void actionPerformed(ActionEvent ae)
   {
      if(ae.getSource( ) == copyBtn)
      {
         outputTfd.setText(inputTar.getSelectedText( ));
         System.out.println(inputTar.getSelectedText( ) + ",");
      }
   }
}
```

**8.1.16. 선택사건이란 무엇이며 어떤 조작이 선택사건을 일으키는가? 선택사건을 만들수 있는 GUI조종부분품들에는 어떤것들이 있으며 그것들사이의 다른 점은 무엇인가? 어떤 장소에 각각 적합한가?**

선택사건은 선택항목을 표시하는 선택상태가 변화를 일으키는 사건이다. 내리떨구기목록(Choice), 목록(List), 검사칸(Checkbox), 검사칸그룹(CheckboxGroup)을 변화시키는 선택상태가 선택사건을 일으킨다.

검사칸은 둘중의 하나만을 선택하는데 적합하다. 즉 《옳다》, 《아니다》중 어느 하나만을 선택한다. 검사칸그룹과 내리떨구기목록은 여러개중에서 하나를 선택하는데 적합하다. 서로 다른점은 검사칸그룹은 모든 선택항목을 대면에 배렬한다는것이다. 내리떨구기목록은 선택된 항목을 제외한 나머지 항목은 숨기므로 선택항목이 비교적 많은데 적합하다. 목록의 외형은 내리떨구기목록과 비슷하지만 여러개중에서 많은것을 선택하는데 적합하다.

**8.1.17. 검사칸을 리용하여 단추의 배경색을 표시하고 검사칸그룹을 리용하여 3가지 서체형식을 표시하며 내리떨구기목록을 리용하여 서체크기를 선택하는 프로그람을 작성하시오. 이 프로그람을 통하여 사용자가 단추의 배경색과 전경글자의 현시효과를 확정하도록 하시오.**

원천프로그람 ch8_e8_17.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_17 extends Applet implements ItemListener
{
   Checkbox ckb = new Checkbox("배경색");
   CheckboxGroup style = new CheckboxGroup( );
   Checkbox p = new Checkbox("보통", true, style);
   Checkbox b = new Checkbox("굵은체", false, style);
   Checkbox i = new Checkbox("경사체", false, style);
   Choice size = new Choice( );
   Button btn = new Button("효과");
   public void init( )
   {
      add(ckb);
      add(p);
      add(b);
      add(i);
```

```
        size.add("10");
        size.add("14");
        size.add("18");
        add(size);
        add(btn);
        p.addItemListener(this);
        b.addItemListener(this);
        i.addItemListener(this);
        ckb.addItemListener(this);
        size.addItemListener(this);
    }
    public void itemStateChanged(ItemEvent e)
    {
        Checkbox temp;
        Font oldF = btn.getFont( );
        String s;
        int si;
        if(e.getItemSelectable( ) instanceof Checkbox)
        {
            temp = (Checkbox)(e.getItemSelectable( ));
            if(temp.getLabel( ) == "배경색")
                if(temp.getState( ))
                    btn.setBackground(Color.cyan);
                else
                    btn.setBackground(Color.gray);
            else if(temp.getLabel( ) == "보통" && temp.getState( ))
                btn.setFont(new Font(oldF.getName( ), Font.PLAIN, oldF.getSize( )));
            else if(temp.getLabel( ) == "굵은체" && temp.getState( ))
                btn.setFont(new Font(oldF.getName( ), Font.BOLD, oldF.getSize( )));
            else if(temp.getLabel( ) == "경사체" && temp.getState( ))
                btn.setFont(new Font(oldF.getName( ), Font.ITALIC, oldF.getSize( )));
        }
        else if(e.getItemSelectable( ) instanceof Choice)
        {
            Choice ctemp = (Choice)(e.getItemSelectable( ));
            s = ctemp.getSelectedItem( );
            si = Integer.parseInt(s);
```

```
            btn.setFont(new Font(oldF.getName( ), oldF.getStyle( ), si));
            repaint( );
        }
    }
}
```

**8.1.18. 조종사건이란 무엇이며 조종사건과 선택사건의 다른 점은 무엇인가? 흘림띠란 무엇이며 흘림띠를 어떻게 만들고 리용하는가? 1개의 흘림띠를 포함하는 Applet를 작성하시오. Applet에서 1개의 원을 그리고 흘림띠가 현시하는 수자를 리용하여 그 원의 직경을 표시하고 사용자가 흘림띠를 움직일 때 원의 크기가 그에 따라 변하게 하시오.**

조종사건은 상태가 련속변화를 일으킬수 있는 사건이며 선택사건은 유한인 몇개 상태중에서 한개를 선택하는 사건이다. 조종사건이 표시하는 상태변화는 련속적이며 선택사건이 포함하는 상태개수보다 훨씬 많다.

```
원천프로그람 ch8_e8_18.java
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_18 extends Applet implements AdjustmentListener
{
   Scrollbar mySlider = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 400);
   int radius = 0;
   int x = 200, y = 200;
   public void init( )
   {
      mySlider.setUnitIncrement(1);
      mySlider.setBlockIncrement(0);
      setLayout(new BorderLayout( ));
      add("North", mySlider);
      mySlider.addAdjustmentListener(this);
   }
   public void paint(Graphics g)
   {
      g.drawOval(x - radius, y - radius, radius * 2, radius * 2);
   }
   public void adjustmentValueChanged(AdjustmentEvent ae)
   {
```

```
            if(ae.getAdjustable( ) == mySlider)
            {
                radius = ae.getValue( ) / 2;
                repaint( );
            }
        }
    }
```

8.1.19. 마우스사건에 응답하는 Applet를 작성하시오. 사용자가 마우스를 이동하는데 따라 Applet에서 직4각형을 그리고 상태띠에 마우스의 현재위치를 현시하시오.

원천프로그람 ch8_e8_19.java

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_19 extends Applet implements MouseListener, MouseMotionListener
{
    int beginX = 0, beginY = 0, endX = 0, endY = 0;
    public void init( )
    {
        this.addMouseListener(this);
        this.addMouseMotionListener(this);
    }
    public void paint(Graphics g)
    {
        int x1, x2, y1, y2;
        if(beginX < endX)
        {
            x1 = beginX;
            x2 = endX;
        }
        else
        {
            x1 = endX;
            x2 = beginX;
        }
        if(beginY < endY)
        {
```

```
            y1 = beginY;
            y2 = endY;
        }
        else
        {
            y1 = endY;
            y2 = beginY;
        }
        g.drawRect(x1, y1, x2 - x1, y2 - y1);
    }
    public void mouseClicked(MouseEvent me)
    {
    }
    public void mouseEntered(MouseEvent me)
    {
    }
    public void mouseExited(MouseEvent me)
    {
    }
    public void mousePressed(MouseEvent me)
    {
        beginX = endX = me.getX( );
        beginY = endY = me.getY( );
    }
    public void mouseReleased(MouseEvent me)
    {
    }
    public void mouseMoved(MouseEvent me)
    {
        showStatus("(" + me.getX( ) + ", " + me.getY( ) + ")");
    }
    public void mouseDragged(MouseEvent me)
    {
        endX = me.getX( );
        endY = me.getY( );
        showStatus("(" + endX + ", " + endY + ")");
        repaint( );
    }
}
```

8.1.20. 문제 8.1.19를 수정하고 하나의 벡토르객체를 리용하여 사용자가 그린 매개 직4각형을 보관하고 현시하며 건반사건에 응답하여 사용자가 건 q를 누를 때 화면의 모든 직4각형을 삭제하시오.

원천프로그람 ch8_e8_20.java

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class ch8_e8_21 extends Applet implements MouseListener,
   MouseMotionListener, KeyListener
{
   Vector myRectVector = new Vector( );
   int beginX = 0, beginY = 0, endX = 0, endY = 0;
   public void init( )
   {
      this.addMouseListener(this);
      this.addMouseMotionListener(this);
      this.addKeyListener(this);
   }
   public void paint(Graphics g)
   {
      int x1, x2, y1, y2;
      if(beginX<endX)
      {
         x1 = beginX;
         x2 = endX;
      }
      else
      {
         x1 = endX;
         x2 = beginX;
      }
      if(beginY < endY)
      {
         y1 = beginY;
         y2 = endY;
```

```
        }
        else
        {
            y1 = endY;
            y2 = beginY;
        }
        g.drawRect(x1, y1, x2 - x1, y2 - y1);
        for(Enumeration e = myRectVector.elements(); e.hasMoreElements();)
        {
            Rectangle rect = (Rectangle)(e.nextElement( ));
            g.drawRect((int)rect.getX( ), (int)rect.getY( ),
                (int)rect.getWidth( ), (int)rect.getHeight( ));
        }
    }
    public void mouseClicked(MouseEvent me)
    {
    }
    public void mouseEntered(MouseEvent me)
    {
    }
    public void mouseExited(MouseEvent me)
    {
    }
    public void mousePressed(MouseEvent me)
    {
        beginX = endX = me.getX( );
        beginY = endY = me.getY( );
    }
    public void mouseReleased(MouseEvent me)
    {
        myRectVector.add(new Rectangle(Math.min(beginX, endX),
            Math.min(beginY, endY), Math.abs(endX-beginX), Math.abs(endY-beginY)));
    }
    public void mouseMoved(MouseEvent me)
    {
        showStatus("(" + me.getX( ) + ", "+me.getY( ) + ")");
    }
    public void mouseDragged(MouseEvent me)
```

```
        {
            endX = me.getX( );
            endY = me.getY( );
            showStatus("(" + endX + ", " + endY + ")");
            repaint( );
        }
        public void keyPressed(KeyEvent ke)
        {
        }
        public void keyReleased(KeyEvent ke)
        {
        }
        public void keyTyped(KeyEvent ke)
        {
            if(ke.getKeyChar( ) == 'q')
            {
                myRectVector.removeAllElements( );
                beginX = endX = 0;
                beginY = endY = 0;
                repaint( );
            }
        }
    }
```

**8.1.21. 문제 8.1.17의 프로그람을 수정하여 하나의 Canvas와 그 우의 문자렬을 리용하여 매개 선택패키지가 확정한 현시효과를 현시하시오.**

원천프로그람 ch8_e8_21.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_21 extends Applet implements ItemListener
{
    Checkbox ckb = new Checkbox("배경색");
    CheckboxGroup style = new CheckboxGroup( );
    Checkbox p = new Checkbox("보통", true, style);
    Checkbox b = new Checkbox("굵은체", false, style);
    Checkbox i = new Checkbox("경사체", false, style);
    Choice size = new Choice( );
```

```
MyCanvas myCanvas = new MyCanvas( );
public void init( )
{
   add(ckb);
   add(p);
   add(b);
   add(i);
   size.add("10");
   size.add("14");
   size.add("18");
   add(size);
   add(myCanvas);
   myCanvas.setSize(new Dimension(400, 200));
   p.addItemListener(this);
   b.addItemListener(this);
   i.addItemListener(this);
   ckb.addItemListener(this);
   size.addItemListener(this);
}
public void itemStateChanged(ItemEvent e)
{
   Checkbox temp;
   Font oldF = myCanvas.getFont( );
   String s;
   int si;
   if(e.getItemSelectable( ) instanceof Checkbox)
   {
      temp = (Checkbox)(e.getItemSelectable( ));
      if(temp.getLabel( ) == "배경색")
         if(temp.getState( ))
            myCanvas.setBackground(Color.cyan);
         else
            myCanvas.setBackground(Color.gray);
      else if(temp.getLabel( ) == "보통" && temp.getState( ))
         myCanvas.setFont(new Font(oldF.getName( ), Font.PLAIN, oldF.getSize( )));
      else if(temp.getLabel( ) == "굵은체" && temp.getState( ))
         myCanvas.setFont(new Font(oldF.getName( ), Font.BOLD, oldF.getSize( )));
```

```
        else if(temp.getLabel( ) == "경사체" && temp.getState( ))
          myCanvas.setFont(new Font(oldF.getName( ), Font.ITALIC, oldF.getSize( )));
        repaint( );
      }
      else if(e.getItemSelectable( ) instanceof Choice)
      {
        Choice ctemp = (Choice)(e.getItemSelectable( ));
        s = ctemp.getSelectedItem( );
        si = Integer.parseInt(s);
        myCanvas.setFont(new Font(oldF.getName( ), oldF.getStyle( ), si));
        repaint( );
      }
    }
  }
  class MyCanvas extends Canvas
  {
    public void paint(Graphics g)
    {
      g.drawString("무엇이 변경되는지 말해보시오.", 10, 40);
    }
  }
```

**8.1.22. 용기의 배치방안이란 무엇이며 Java프로그람에서 자주 리용하는 배치방안에는 어떤것이 있는가를 서술하시오.**

용기의 배치방안은 용기에 놓이게 될 부분품의 현시위치관계를 표현한것을 말한다.

Java에서 자주 리용하는 배치방안에는 FlowLayout, BorderLayout, CardLayout, GridLayout, GridBagLayout가 있다.

**8.1.23. 계수기를 실현하는 Applet프로그람을 작성하시오. 그중에 0 - 9까지의 10개수자단추, 더하기, 덜기, 곱하기, 나누기단추 및 같기부호를 포함하며 빈 2개의 보조단추를 포함한다. 또한 입출력을 현시하는 본문마당이 있다. 각각 BorderLayout와 GridLayout를 리용하여 실현하시오.**

```
원천프로그람 ch8_e8_23.java
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_23 extends Applet implements ActionListener
```

```
{
    double firstOperant, secondOperant;
    int caculateStatus = 0;
    char operator;
    Button caculateBtn[ ] = new Button[16];
    String btnLabel[ ]={"1", "2", "3", "+", "4", "5", "6", "-", "7", "8", "9", "*", "0", "c", "=", "/"};
    TextField inOutTfd = new TextField(50);
    Panel btnPanel = new Panel( );
    public void init( )
    {
        btnPanel.setLayout(new GridLayout(4, 4));
        for(int i = 0; i < 16; i++)
        {
            caculateBtn[i] = new Button(btnLabel[i]);
            caculateBtn[i].addActionListener(this);
            btnPanel.add(caculateBtn[i]);
        }
        setLayout(new BorderLayout( ));
        add("North", inOutTfd);
        add("Center", btnPanel);
        inOutTfd.setText("0");
        caculateStatus = 0;
    }
    public void actionPerformed(ActionEvent ae)
    {
        if(ae.getSource( ) instanceof Button)
        {
            char ch = ((Button)(ae.getSource( ))).getLabel( ).charAt(0);
            //System.out.println(ch + ", status: " + caculateStatus);
            switch(ch)
            {
                case 'c':
                    inOutTfd.setText("0");
                    firstOperant = secondOperant=0;
                    caculateStatus = 0;
                    break;
                case '=':
```

```
        if(caculateStatus == 0)
        {
            caculateStatus = 3;
        }
        else if(caculateStatus == 2)
        {
            oneStepCaculate( );
            //System.out.println(firstOperant + ", " + secondOperant);
            caculateStatus = 3;
        }
        break;
    case '+':
    case '-':
    case '*':
    case '/':
        if(caculateStatus == 0 | | caculateStatus == 3)
        {
            secondOperant = 0;
            caculateStatus = 1;
        }
        else if(caculateStatus == 1)
        {
            operator = ch;
        }
        else if(caculateStatus == 2)
        {
            oneStepCaculate( );
            secondOperant = 0;
            caculateStatus = 1;
        }
        operator = ch;
        break;
    default: //수자건
        if(caculateStatus == 0)
        {
            int temp = (int)firstOperant;
            temp = temp * 10 + ch - '0';
```

```
                inOutTfd.setText(Integer.toString(temp));
                firstOperant = temp;
            }
            else if(caculateStatus == 1)
            {
                secondOperant = ch - '0';
                inOutTfd.setText(Integer.toString((int)secondOperant));
                caculateStatus = 2;
            }
            else if(caculateStatus == 2)
            {
                int temp = (int)secondOperant;
                temp = temp * 10 + ch - '0';
                inOutTfd.setText(Integer.toString(temp));
                secondOperant = temp;
            }
            else if(caculateStatus == 3)
            {
                firstOperant = ch - '0';
                inOutTfd.setText(Integer.toString((int)firstOperant));
                secondOperant = 0;
                caculateStatus = 0;
            }
        }//switch
    }
}
public void oneStepCaculate( )
{
    switch(operator)
    {
        case '+':
            firstOperant += secondOperant;
            break;
        case '-':
            firstOperant -= secondOperant;
            break;
        case '*':
```

```
                firstOperant *= secondOperant;
                break;
            case '/':
                firstOperant /= secondOperant;
                break;
        }
        inOutTfd.setText(Double.toString(firstOperant));
    }
}
```

**8.1.24. Panel과 Applet는 어떤 관계가 있는가? Panel은 Java프로그람에서 어떤 작용을 하는가?**

Applet는 Panel의 하위클라스이다. Panel은 항상 몇개의 부분품을 포함하며 그것들을 하나로 다른 용기에 추가할수 있다. 이렇게 매개 부분품의 층차적인 조직을 실현한다.

**8.1.25. Frame을 왜 매우 중요한 용기라고 하는가? 이것을 리용하는 프로그람이 항상 WindowListener를 실현해야 하는 리유는 무엇이며 이것을 닫는데 어떠한 방법들이 있는가?**

Frame은 독립적으로 존재할수 있고 테두리가 있는 제일 바깥층의 용기이다. 다른 임의의 용기에 놓을수 없다.

사용자가 그림기호를 눌러 Freme창문을 닫는 명령에 응답하기 위하여서는 반드시 WindowListener대면을 실현하여야 한다.

닫는 방법에는 3가지가 있다. 전용단추설치, 차림표명령정의, 자체가 가지고있는 조종기호를 리용하는 방법이다.

**8.1.26. 문제 8.1.14를 도형대면의 Application프로그람으로 고치시오.**

```
원천프로그람 ch8_e8_26.java
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_26
{
    public static void main(String args[ ])
    {
        new MyFrame( );
    }
}
class MyFrame extends Frame implements ActionListener,WindowListener
```

```
{
    Label outputLbl = new Label(" ");
    TextField inputTfd = new TextField(10);
    Button copyBtn = new Button("복사");
    MyFrame( )
    {
        super("나의 창문");
        inputTfd.setText(" ");
        outputLbl.setText(" ");
        setLayout(new FlowLayout( ));
        add(inputTfd);
        add(copyBtn);
        add(outputLbl);
        copyBtn.addActionListener(this);
        addWindowListener(this);
        setSize(300, 200);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent ae)
    {
        if(ae.getSource( ) == copyBtn)
        {
            outputLbl.setText(inputTfd.getText( ));
        }
    }
    public void windowClosing(WindowEvent we)
    {
        dispose( );
        System.exit(0);
    }
    public void windowActivated(WindowEvent we)
    {
    }
    public void windowClosed(WindowEvent we)
    {
    }
    public void windowDeactivated(WindowEvent we)
```

```
        {
        }
        public void windowOpened(WindowEvent we)
        {
        }
        public void windowIconified(WindowEvent we)
        {
        }
        public void windowDeiconified(WindowEvent we)
        {
        }
    }
```

**8.1.27.** Canvas객체를 리용하여 문제 8.1.18의 프로그람을 도형대면의 Application프로그람으로 고치시오.

```
원천프로그람 ch8_e8_27.java
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_27
{
    public static void main(String args[ ])
    {
        new MyFrame( );
    }
}
class MyFrame extends Frame implements AdjustmentListener
{
    Scrollbar mySlider = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 400);
    int radius = 0;
    int x = 200, y = 200;
    MyFrame( )
    {
        super("나의 창문");
        mySlider.setUnitIncrement(1);
        mySlider.setBlockIncrement(10);
        setLayout(new BorderLayout( ));
        add("North", mySlider);
```

```
        mySlider.addAdjustmentListener(this);
        addWindowListener(new closeWin( ));
        setSize(650, 400);
        setVisible(true);
    }
    public void paint(Graphics g)
    {
        g.drawOval(x - radius, y - radius, radius * 2, radius * 2);
    }
    public void adjustmentValueChanged(AdjustmentEvent ae)
    {
        if(ae.getAdjustable( ) == mySlider)
        {
            radius = ae.getValue( ) / 2;
            repaint( );
        }
    }
}
class closeWin extends WindowAdapter
{
    public void windowClosing(WindowEvent we)
    {
        Frame frm = (Frame)we.getWindow( );
        frm.dispose( );
        System.exit(0);
    }
}
```

**8.1.28. 자주 리용하는 차림표에는 어떤것이 있는가? 임의의 용기가 다 차림표를 리용할수 있는가? 차림표를 실현하는 프로그람작성순서를 간단히 서술하시오.**

Java프로그람에서 자주 리용하는 차림표에는 차림표띠형식의 차림표와 튀여나오기차림표가 있다.

오직 MenuContainer대면을 실현한 용기에서만 차림표띠형식의 차림표를 리용할수 있으며 MouseListener대면을 실현한 용기나 부품에서만 튀여나오기차림표를 리용할수 있다.

차림표띠형식의 차림표를 실현하는 프로그람작성순서는 차림표만들기, 차림표항목만들기, 차림표부분항목만들기이다. 차림표를 MenuContianer대면을 실현하는 용기에 추가

하고 차림표부분항목을 ActionListener대면을 실현하는 동작사건감시자에게 등록한다. 서로 다른 차림표부분항목에 대응하는 서로 다른 구체적인 조작을 정의한다.

튀여나오기차림표를 실현하는 구체적인 프로그람작성순서는 다음과 같다.

튀여나오기차림표만들기, 차림표부분항목만들기, 차림표부분항목등록하기, 구체적인 조작을 대응시키는것이다. 튀여나오기차림표를 지정된 부분품에 추가하고 그 부분품에서 마우스사건에 대한 응답을 정의하여 마우스누르기위치에서 튀여나오기차림표를 현시하도록 한다.

**8.1.29. 하나의 차림표를 포함하며 이 차림표의 탈퇴를 선택하여 Application의 창문을 닫고 프로그람을 결속하는 도형대면의 Application프로그람을 작성하시오.**

원천프로그람 ch8_e8_29.java

```
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_29
{
   public static void main(String args[ ])
   {
      new MyFrame( );
   }
}
class MyFrame extends Frame implements ActionListener
{
   MenuBar m_MenuBar;
   Menu menuFile;
   MenuItem mi_File_Exit;
   MyFrame( )
   {
      super("나의 창문");
      m_MenuBar = new MenuBar( );
      menuFile = new Menu("파일");
      mi_File_Exit = new MenuItem("탈퇴");
      mi_File_Exit.setActionCommand("탈퇴");
      mi_File_Exit.addActionListener(this);
      menuFile.add(mi_File_Exit);
      m_MenuBar.add(menuFile);
      this.setMenuBar(m_MenuBar);
```

```
        this.addWindowListener(new closeWindow( ));
        setSize(300, 200);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent ae)
    {
        if(ae.getActionCommand( ).equals("탈퇴"))
        {
            dispose( );
            System.exit(0);
        }
    }
}
class closeWindow extends WindowAdapter
{
    public void windowClosing(WindowEvent we)
    {
        Frame frm=(Frame)we.getWindow( );
        frm.dispose( );
        System.exit(0);
    }
}
```

**8.1.30. Application창문에 하나의 본문구역과 하나의 단추를 포함하며 본문구역에 미리 한 단락의 문장이 포함되여있다. 초점감시자와 본문감시자를 리용하여 아래의 기능을 실현하도록 문제 8.1.29의 프로그람을 보충하시오.**

사용자가 본문마당의 내용을 수정하고 이 본문마당을 탈퇴할 때 프로그람은 사용자로 하여금 튀여나오기대화칸을 확인수정하게 한다. 만일 사용자가 이 본문마당에 들어가 내용을 수정하지 않고 탈퇴한다면 대화칸이 튀여나오지 않는다.

```
원천프로그람 ch8_e8_30.java
import java.awt.*;import java.awt.event.*;
public class ch8_e8_30
{
    public static void main(String args[ ])
    {
        new MyFrame( );
    }
```

```
}
class MyFrame extends Frame implements ActionListener, FocusListener, TextListener
{
   TextArea myTextArea;
   MenuBar m_MenuBar;
   Menu menuFile;
   MenuItem mi_File_Exit;
   boolean textModified=false;
   String orgMsg = "본래정보";
   Dialog modifyConfirmDialog;
   Button confirmBtn, denyBtn, otherBtn;
   MyFrame( )
   {
      super("나의 창문");
      myTextArea = new TextArea(10, 45);
      myTextArea.setText(orgMsg);
      myTextArea.addTextListener(this);
      myTextArea.addFocusListener(this);
      confirmBtn = new Button("예");
      denyBtn = new Button("아니");
      confirmBtn.addActionListener(this);
      denyBtn.addActionListener(this);
      m_MenuBar = new MenuBar( );
      menuFile = new Menu("파일");
      mi_File_Exit = new MenuItem("탈퇴");
      mi_File_Exit.setActionCommand("탈퇴");
      mi_File_Exit.addActionListener(this);
      menuFile.add(mi_File_Exit);
      m_MenuBar.add(menuFile);
      this.setMenuBar(m_MenuBar);
      addWindowListener(new closeWindow( ));
      otherBtn = new Button("확인");
      setLayout(new FlowLayout( ));
      add(myTextArea);
      add(otherBtn);
      pack( );
      show( );
   }
```

```
public void focusGained(FocusEvent fe)
{
   textModified = false;
}
public void focusLost(FocusEvent fe)
{
   if(textModified)
   {
      modifyConfirmDialog = new Dialog(this, "확인수정", true);
      modifyConfirmDialog.setLayout(new FlowLayout( ));
      modifyConfirmDialog.add(new Label(
      "본문내용이 수정되였습니다. 보관하시겠습니까?"));
      modifyConfirmDialog.add(confirmBtn);
      modifyConfirmDialog.add(denyBtn);
      modifyConfirmDialog.pack( );
      modifyConfirmDialog.show( );
   }
}
public void textValueChanged(TextEvent te)
{
   if(te.getSource( ) == myTextArea)
   {
      textModified = true;
   }
}
public void actionPerformed(ActionEvent ae)
{
   if(ae.getSource( ) == confirmBtn)
   {
      orgMsg = myTextArea.getText( );
      textModified = false;
      modifyConfirmDialog.dispose( );
   }
   else if(ae.getSource( ) == denyBtn)
   {
      myTextArea.setText(orgMsg);
      textModified = false;
```

```
            modifyConfirmDialog.dispose( );
        }
        else if(ae.getActionCommand( ).equals("탈퇴"))
        {
            dispose( );
            System.exit(0);
        }
    }
}
class closeWindow extends WindowAdapter
{
    public void windowClosing(WindowEvent we)
    {
        Frame frm = (Frame)we.getWindow( );
        frm.dispose( );
        System.exit(0);
    }
}
```

**8.1.31. 이 장에서 학습한 내용에 근거하여 Java Application으로 모의문자편집기를 작성하시오.**

원천프로그람 ch8_e8_31.java

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class ch8_e8_31
{
    public static void main(String args[ ])
    {
        MySimpleTextEditor editor = new MySimpleTextEditor( );
        editor.setSize(300, 200);
        editor.setVisible(true);
        editor.addWindowListener(new WindowAdapter( )
        {
            public void windowClosing(WindowEvent we)
            {
                System.exit(0);
```

```
            }
        });
    }
}
class MySimpleTextEditor extends Frame implements ActionListener
{
    private MenuBar myBar = new MenuBar( );
    private Menu fileMenu, editMenu, cutsMenu;
    private MenuItem cutItem, copyItem, pasteItem, selectAllItem;
    private MenuItem recentcutItem;
    private MenuItem quitItem, openItem, saveItem;
    private TextArea mainWorkWindow = new TextArea(10, 15);
    private String placeHolder = " ";
    private Vector recentCuts = new Vector( );
    public MySimpleTextEditor( )
    {
        super("나의 본문편집기");
        setLayout(new BorderLayout( ));
        add("Center", mainWorkWindow);
        setMenuBar(myBar);
        initFileMenu( );
        initEditMenu( );
    }
    private void initEditMenu( )
    {
        editMenu = new Menu("편집");
        myBar.add(editMenu);
        cutItem = new MenuItem("자르기");
        cutItem.addActionListener(this);
        editMenu.add(cutItem);
        copyItem = new MenuItem("복사");
        copyItem.addActionListener(this);
        editMenu.add(copyItem);
        pasteItem = new MenuItem("붙이기");
        pasteItem.addActionListener(this);
        editMenu.add(pasteItem);
        editMenu.addSeparator( );
```

```
    selectAllItem = new MenuItem("모두 선택");
    selectAllItem.addActionListener(this);
    editMenu.add(selectAllItem);
    editMenu.addSeparator( );
    cutsMenu = new Menu("오려붙이기");
    editMenu.add(cutsMenu);
}
private void initFileMenu( )
{
    fileMenu = new Menu("파일");
    myBar.add(fileMenu);
    openItem = new MenuItem("열기");
    openItem.addActionListener(this);
    openItem.setEnabled(false);
    fileMenu.add(openItem);
    saveItem = new MenuItem("보관");
    saveItem.addActionListener(this);
    saveItem.setEnabled(false);
    fileMenu.add(saveItem);
    fileMenu.addSeparator( );
    quitItem = new MenuItem("탈퇴");
    quitItem.addActionListener(this);
    fileMenu.add(quitItem);
}
public void actionPerformed(ActionEvent e)
{
    if(!(e.getSource( ) instanceof MenuItem))
    return;
    MenuItem m = (MenuItem)e.getSource( );
    if(m == quitItem)
    {
        dispose( );
        System.exit(0);
    }
    else if(m == cutItem)
    {
        placeHolder = mainWorkWindow.getSelectedText( );
```

```
                mainWorkWindow.replaceRange(" ", mainWorkWindow.getSelectionStart(),
                    mainWorkWindow.getSelectionEnd());
                addRecentCut(placeHolder);
            }
            else if(m == copyItem)
            {
                placeHolder = mainWorkWindow.getSelectedText();
            }
            else if(m == pasteItem)
            {
                mainWorkWindow.insert(placeHolder, mainWorkWindow.getCaretPosition());
            }
            else if(m == selectAllItem)
            {
                mainWorkWindow.selectAll();
            }
        }
        private void addRecentCut(String cut)
        {
            recentCuts.insertElementAt(cut, 0);
            cutsMenu.removeAll();
            for(int i = 0; i < recentCuts.size(); i++)
            {
                MenuItem item = new MenuItem((String)recentCuts.elementAt(i));
                cutsMenu.add(item);
            }
        }
    }
```

**8.1.32. Java프로그람에서 Swing GUI부분품을 리용하려면 프로그람의 시작부분에 어떤 패키지를 추가하여야 하는가?**

javax.swing패키지를 추가하여야 한다.

**8.1.33. JApplet와 Applet는 어떻게 서로 다른가? JApplet는 어떤 기정배치방안을 리용하며 거기에 어떻게 Swing GUI부분품을 추가하는가?**

Japplet의 기정배치방안은 BorderLayout이며 Applet의 기정배치방안은 FlowLayout이다. Applet에서 부분품을 추가하려면 Applet자체의 add( )메쏘드를 직접 리용할수 있고 JApplet에서 부분품을 추가하려면 반드시 getContentPane( )으로 현재 JApplet의 용기를 얻은 다음 그 용기객체의 add( )메쏘드를 리용하여야 한다.

**8.1.34.** JLabel객체를 포함하며 자기의 이름을 현시하는 JApplet프로그람을 작성하시오.

```
원천프로그람ch8_e8_34.java
import javax.swing.*;
import java.awt.*;
public class ch8_e8_34 extends JApplet
{
   JLabel myLbl = new JLabel("My Name");
   public void init( )
   {
      Container c = getContentPane( );
      c.add("North", myLbl);
   }
}
```

**8.1.35.** JButton과 Button의 다른 점은 무엇인가? 그림기호를 가지고있는 JButton객체를 포함하며 사용자가 이 단추를 누를 때 Application프로그람이 그 Frame의 표제를 《단추누르기》로 고치는 도형대면의 Application프로그람을 작성하시오.

Button에 비하여 JButton에는 많은 실용적인 기능이 추가되여있다. 실례로 단추에 그림기호의 추가, 마우스의 끌기 혹은 누르기할 때 서로 다른 효과의 현시, 단추기능제시 등이다.

```
원천프로그람 ch8_e8_35.java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_35
{
   public static void main(String args[ ])
   {
      new MyFrame( );
   }
}
class MyFrame extends Frame implements ActionListener
{
   JButton myButton = new JButton("Change It!");
   MyFrame( )
```

```
    {
      super("나의 창문");
      add("North", myButton);
      myButton.addActionListener(this);
      addWindowListener(new closeWin( ));
      setSize(300, 200);
      setVisible(true);
    }
    public void actionPerformed(ActionEvent ae)
    {
      if(ae.getSource( ) == myButton)
      {
        setTitle("단추누르기");
      }
    }
}
class closeWin extends WindowAdapter
{
    public void windowClosing(WindowEvent we)
    {
      we.getWindow( ).dispose( );
      System.exit(0);
    }
}
```

**8.1.36. 문제 8.1.35에 기초하여 단추에서 마우스를 누르거나 끌기할 때 그 그림기호가 서로 다른 효과를 가지도록 프로그람을 수정하시오.**

```
원천프로그람 ch8_e8_36.java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ch8_e8_36
{
    public static void main(String args[ ])
    {
      new MyFrame( );
    }
}
```

```
class MyFrame extends Frame implements ActionListener
{
   JButton myButton = new JButton(new ImageIcon("normal.gif"));
   MyFrame( )
   {
      super("나의 창문");
      myButton.setPressedIcon(new ImageIcon("pressed.gif"));
      myButton.setRolloverIcon(new ImageIcon("rollover.gif"));
      myButton.setRolloverEnabled(true);
      add("North", myButton);
      myButton.addActionListener(this);
      addWindowListener(new closeWin( ));
      setSize(300, 200);
      setVisible(true);
   }
   public void actionPerformed(ActionEvent ae)
   {
      if(ae.getSource( ) == myButton)
      {
         setTitle("단추누르기");
      }
   }
}
class closeWin extends WindowAdapter
{
   public void windowClosing(WindowEvent we)
   {
      we.getWindow( ).dispose( );
      System.exit(0);
   }
}
```

**8.1.37. 문제 8.1.36의 단추로 제시정보 《change》를 추가하시오.**

원천프로그람 ch8_e8_37.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

```
public class ch8_e8_37 extends JApplet implements ActionListener
{
   JButton myButton = new JButton(new ImageIcon("normal.gif"));
   public void init( )
   {
      myButton.setPressedIcon(new ImageIcon("pressed.gif"));
      myButton.setRolloverIcon(new ImageIcon("rollover.gif"));
      myButton.setRolloverEnabled(true);
      myButton.setToolTipText("change");
      getContentPane( ).add("North", myButton);
      myButton.addActionListener(this);
   }
   public void actionPerformed(ActionEvent ae)
   {
      if(ae.getSource( ) == myButton)
      {
         showStatus("단추누르기");
      }
   }
}
```

**8.1.38. JSlider와 Scrollbar의 다른 점은 무엇인가? 3개의 JSlider와 한개의 Jlabel객체를 포함하는 Applet프로그람을 작성하시오. 3개의 흘림띠는 각각 붉은색, 록색, 청색의 3가지 색의 비례를 조절하는데 리용한다. 매개 JSlider는 0~255로 된 자의 눈금을 표시한다.(눈금간격을 자유롭게 정할수 있다) 사용자가 흘림띠를 이동하여 3색의 비례를 수정하는데 맞게 JLabel의 배경색을 수정한다.**

JSlider와 Scrollbar의 기능은 대체로 같지만 그것들이 일으키는 사건이름은 서로 다르다. JSlider에는 눈금현시기능이 첨부되여있다.

```
원천프로그람 ch8_e8_38.java
import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;
public class ch8_e8_38 extends JApplet implements ChangeListener
{
   JButton myButton = new JButton("My True Color");
   JSlider redSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 204);
   JSlider greenSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 204);
   JSlider blueSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 204);
```

```
    int value=0;
    public void init( )
    {
       Panel p = new Panel( );
       p.setLayout(new GridLayout(3, 1));
       p.add(redSlider);
       p.add(greenSlider);
       p.add(blueSlider);
       redSlider.addChangeListener(this);
       greenSlider.addChangeListener(this);
       blueSlider.addChangeListener(this);
       getContentPane( ).add("East", myButton);
       getContentPane( ).add("Center", p);
    }
    public void stateChanged(ChangeEvent ae)
    {
       Color oldColor=myButton.getBackground( );
       if(ae.getSource( ) == redSlider)
       {
          value=redSlider.getValue( );
          myButton.setBackground(new Color(value,
          oldColor.getGreen( ), oldColor.getBlue( )));
       }
       else if(ae.getSource( ) == greenSlider)
       {
          value=greenSlider.getValue( );
          myButton.setBackground(new Color(oldColor.getRed( ),
          value, oldColor.getBlue( )));
       }
       else if(ae.getSource( ) == blueSlider)
       {
          value=blueSlider.getValue( );
          myButton.setBackground(new Color(oldColor.getRed( ),
          oldColor.getGreen( ), value));
       }
    }
}
```

**8.1.39. JPasswordField는 어느것의 하위클라스이며 어떤 특징이 있는가? 사용자가 입력한 대장번호와 암호를 접수하고 검증하는 JApplet프로그람을 작성하시오.**

JPasswordField는 JTextField의 하위클라스이며 그 특징은 사용자가 입력한 내용을 숨길수 있는것이다.

원천프로그람 ch8_e8_39.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
public class ch8_e8_39 extends JApplet implements ActionListener
{
   final String SETUP_PROMPT = "Please register:";
   final String SIGNIN_PROMPT = "Please sign up:";
   JLabel myLbl = new JLabel(SETUP_PROMPT);
   JTextField myTfd = new JTextField(10);
   JPasswordField myPwdFd = new JPasswordField(10);
   int count = 0;
   String accountNumber = null;
   String inputAccNum = null;
   String password = null;
   public void init( )
   {
      Container c = getContentPane( );
      c.setLayout(new FlowLayout( ));
      c.add(myLbl);
      c.add(myTfd);
      c.add(myPwdFd);
      myTfd.addActionListener(this);
      myPwdFd.addActionListener(this);
   }
   public void actionPerformed(ActionEvent ae)
   {
      if(ae.getSource( ) == myTfd)
      {
         inputAccNum = myTfd.getText( ).trim( );
         myPwdFd.requestFocus( );
```

```
      System.out.println(inputAccNum+"," + count);
   }
   else if(ae.getSource( ) == myPwdFd)
   {
      if(myLbl.getText( ).equals(SETUP_PROMPT))
      {
         accountNumber = inputAccNum;
         password = String.valueOf(myPwdFd.getPassword( )).trim( );
         myTfd.setText(" ");
         myPwdFd.setText(" ");
         myLbl.setText(SIGNIN_PROMPT);
         myTfd.requestFocus( );
         System.out.println(password + "." + count);
      }
      else if(myLbl.getText( ).equals(SIGNIN_PROMPT))
      {
         String str = String.valueOf(myPwdFd.getPassword( )).trim( );
         System.out.println(inputAccNum + "," + str + "." + count);
         if(inputAccNum.equals(accountNumber) && str.equals(password))
         {
            count = 0;
            myLbl.setText(SETUP_PROMPT);
            showStatus("암호확인성공");
            myTfd.setText(" ");
            myPwdFd.setText(" ");
            myTfd.requestFocus( );
         }
         else
         {
            if(++count < 3)
            {
               myPwdFd.setText(" ");
               showStatus("대장번호 또는 암호입력이 실패하면 다시 입력");
            }
            else
            {
               showStatus("입력이 3번 실패하였습니다. 다시 만납시다.");
```

```
                        myTfd.setEnabled(false);
                        myPwdFd.setEnabled(false);
                    }
                }
            }
        }
    }
}
```

**8.1.40.** JTabbedPane은 CardLayout를 리용한 용기와 어떻게 다른가? JTabbedPane을 포함하며 그의 리용방법을 검증하는 JApplet프로그람을 작성하시오.

JTabbedPane의 배치방안은 CardLayout를 리용한 용기의 배치방안과 류사하지만 JTabbedPane에서 매개 페지는 다 페지표시를 가지고있어 사용자가 그 페지에 들어있는 내용을 료해하는데 편리하다.

원천프로그람 ch8_e8_40.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
public class ch8_e8_40 extends JApplet implements ChangeListener
{
    JTabbedPane myJTabPane = new JTabbedPane( );
    public void init( )
    {
        myJTabPane.add("1", new JLabel("첫번째 페지"));
        myJTabPane.add("2", new JLabel("두번째 페지"));
        myJTabPane.addChangeListener(this);
        getContentPane( ).add("Center", myJTabPane);
    }
    public void stateChanged(ChangeEvent ce)
    {
        if(ce.getSource( ) == myJTabPane)
        {
            int i = ((JTabbedPane)ce.getSource( )).getSelectedIndex( );
            showStatus((i + 1) + "페지를 선택");
        }
    }
}
```

# 8.2. 보충문제

8.2.1. 타원, 원, 직4각형, 3각형을 그릴 때 리용하려는 java.awt패키지의 클라스이름을 쓰시오.

8.2.2. 체계클라스 Color에서는 몇가지 색을 미리 정의하였는가? 그것들을 쓰시오.

8.2.3. g를 Graphics객체라고 할 때 아래의 문장을 읽고 그의 출력결과를 지적하시오.

(1) g.fillRect(100, 100, 80, 120);

(2) g.drawOval(0, 0, 100, 100);

(3) g.drawRoundRect(0, 0, 100, 100, 100, 100);

(4) g.drawString(3 * 5 + " ", 10, 20);

8.2.4. 어떤 가정의 3월분 지출비용이 아래와 같다.

| 살림집 | 교통 | 식품 | 옷 | 교육 | 오락 | 저금 |
|---|---|---|---|---|---|---|
| 1000 | 100 | 500 | 150 | 200 | 80 | 300 |

가정의 이달 지출을 분류하는 기둥도표와 원형도표를 그리는 프로그람을 작성하시오.

8.2.5. 어떤 실험자료곡선도표를 그리는 Java메쏘드를 작성하시오. 형식파라메터로는 Graphics객체인 g와 실험실자료를 포함한 Point배렬이 속하며 메쏘드는 되돌이값을 가지지 않는다. 매개 자료가 가지고있는 순서로 이루어지는 곡선도표를 그리시오.

8.2.6. 아래의 프로그람을 읽고 틀린것을 지적하고 고치시오.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class MyApplet extends Applet implements ActionListener
{
   Button myBtn;
   public void init( )
   {
      Button myBtn = new Button("Click Me!");
   }
   public void actionPerformed(ActionEvent ae)
   {
      if(e.getSource( ) == myBtn)
         myBtn.setLabel("I'm clicked!");
   }
}
```

8.2.7. 섭씨온도와 화씨온도사이의 전환을 실현하는 도형대면의 Java프로그람을 작성하시오. 그 전환공식은 화씨온도=(섭씨온도-32)/1.8이다.

2개의 본문마당으로 사용자가 입력한 자료를 받는다. 3가지 사건응답방법을 찾아내시오. 아래의 어떤 방법을 리용하는것이 제일 편리한가를 비교하시오.

단추동작사건, 본문마당동작사건, 본문사건, 초점사건, 차림표동작사건

8.2.8. 아래의 프로그람이 어떤 대면을 실현하는가? 어떻게 호상작용하는가?

```
import java.awt.*;
import java.awt.event.*;
public class sup_8_1
{
   public static void main(String args[ ])
   {
      new MoveBallFrame( );
   }
}
class MoveBallFrame extends Frame implements AdjustmentListener
{
   Scrollbar myScrollbar = null;
   Point center = null;
   final int RADIUS = 10;
   MoveBallFrame( )
   {
      super("나의 창문");
      center = new Point(100,100);
      myScrollbar = new Scrollbar(Scrollbar.VERTICAL, (int)center.getY( ),
         5, 0, (int)center.getY( ) * 2);
      add("East", myScrollbar);
      myScrollbar.addAdjustmentListener(this);
      addWindowListener(new winClose( ));
      setSize(300, 200);
      setVisible(true);
   }
   public void adjustmentValueChanged(AdjustmentEvent ce)
   {
      if(ce.getSource( ) == myScrollbar)
      {
```

```
            center.setLocation(center.getX( ), ce.getValue( ));
            repaint( );
        }
    }
    public void paint(Graphics g)
    {
        g.fillOval((int)center.getX( ) - RADIUS, (int)center.getY( ) - RADIUS, RADIUS, RADIUS);
    }
}
class winClose extends WindowAdapter
{
    public void windowClosing(WindowEvent we)
    {
        we.getWindow( ).dispose( );
        System.exit(0);
    }
}
```

8.2.9. MouseMotionListener와 MouseActionListener는 어떻게 다른가?

8.2.10. 하나의 3차원적인 원을 우연적으로 그리는 프로그람을 작성하시오. 사용자가 마우스로 이 원을 누를 때 원이 없어지고 화면의 다른 우연위치로 이동하면 다시 현시한다.

8.2.11. 우의 문제를 흥미를 끄는 유희로 고치시오. 사용자의 마우스가 목표원에 다가갈 때 원은 즉시 화면의 다른 위치로 달아난다.

8.2.12. 사용자의 마우스동작에 응답하는 Java프로그람을 작성하시오. 마우스의 누르기위치가 중심으로 될 때 색과 크기가 우연적으로 변하는 원을 그리시오. 마우스를 두번 누를 때 화면은 없어진다.

8.2.13. 화면상에서 우연적으로 색이 변하는 정4각형을 우연적으로 그리는 도형대면의 Java프로그람을 작성하시오. 사용자의 마우스가 이 정4각형안에 들어갈 때 색이 진하게 변하며 마우스가 정4각형밖으로 나오면 색이 본래의 색으로 돌아간다.

8.2.14. WindowListener대면을 실현하는 매개 메쏘드의 메쏘드본체안에서 구체적인 창문사건을 인쇄하는 프로그람을 작성하시오. 실례로 사용자가 창문을 최소로 하면 《창문사건:창문을 최소화》를 인쇄한다. 류사하게 프로그람을 실행하여 Frame창문을 조작하고 매개 사건이 발생하는 상태를 고찰하시오.

8.2.15.뽈이 튀는 유희를 실현하는 Java Application 또는 Java Applet프로그람을 작성하시오. 화면의 임의의 위치에 크기가 고정된 뽈이 생기고 뽈이 우연적인 방향의 직

선을 따라 등속운동한다. 뽈이 벽과 부딪친후 반사각을 따라 반대로 튀여나며 새로운 방향을 따라 다시 벽과 부딪칠 때까지 전진한다. 만일 나머지 힘이 있으면 뽈의 운동속도변화를 더 고려할수 있다. 실례로 벽과 부딪치여 반대로 튀여나오는것은 비교적 빠르고 점차적으로 속도가 감소되며 또는 중력의 영향을 받는다.

8.2.16. 뽈던지기유희를 실현하는 Java프로그람을 작성하시오. 수직흘림띠를 리용하여 뽈이 튀여나오는 각도를 확정하고 각도를 조절한 다음 단추를 눌러 뽈을 던지면 포물선을 그린다.

8.2.17. 우의 문제에 기초하여 화면에 롱구그물을 우연적으로 만들고 각도를 조절하여 매 사람이 매 판에 10개의 뽈을 던질 때 그물에 들어가는 점수를 기록하시오.

8.2.18. 도형대면의 Java프로그람을 만드시오. 도형대면의 TextArea에서 먼저 기록된 문자를 현시한다. 사용자는 TextField 《찾기》에 탐색하려는 문자를 입력하며 TextField 《치환》에 바꾸려는 문자를 입력한다. 《갈아넣기》단추를 눌러 TextArea의 모든 문자를 바꾸어놓는다.

8.2.19. 십자길에서의 교통규정은 《먼저 도착한 차가 먼저 간다.》고 되여있다. 매 차가 십자도로에 도착하면 차를 멈추고 살펴본다. 만일 차가 도착했을 때 다른 길에 기다리는 차가 없으면 계속 앞으로 전진할수 있고 다른 길에 이미 기다리는 차가 있으면 이 차들이 통과하기를 기다렸다가 통과한 후 다시 앞으로 전진해야 한다. 또한 달리는 차와 같은 방향에 기다리는 차대렬이 있으면 그 차가 십자길에 도착하는 시각은 기다리던 차들이 통과한 후 차가 대오의 맨 앞위치에 도착한 시각이다. 1개의 클라스를 리용하여 우연적인 차흐름을 만드는 기능을 실현하시오. 우연적인 시간동안에 한 방향에 차들을 만든다. 이러한 교통상태를 보여주는 도형대면을 설계하고 실현하시오.

8.2.20. 우의 문제를 수정하되 우연적으로 차들을 만드는 4개의 처리를 리용하시오. 매개 처리는 한 방향의 차흐름을 표시한다. 그때 두대 혹은 여러대의 차가 동시에 십자길에 도착할수 있으며 어떤 한 차가 먼저 통과하도록 우연적으로 선택할수 있다.

8.2.21. 아래의 코드를 읽고 관계되는 자료를 자체로 찾아보시오.

만일 명령 while(changeGrayToRed(selected));를 while(!changeGrayToRed(selected));로 고치면 실행효과는 어떻게 다른가?

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
public class sup_8_2
{
   public static void main(String args[ ])
   {
      new GameFrame( );
```

```
    }
}
class GameFrame extends Frame implements ActionListener, Runnable
{
    final int TIME_INTERVAL = 800;
    final long GAME_TIME = 50000;
    int score = 0;
    boolean stoped = false;
    Button buttonArray[ ] = new Button[9];
    Timer gameTimeTimer = null;
    GameTimeTimerTask gameTimeTimerTask = null;
    Thread changeColorThread = null;
    GameFrame( )
    {
        super("붉은색단추를 누르면 1점을 얻을수 있다. 현재 얻은 점수: 0");
        setLayout(new GridLayout(3, 3));
        for(int i = 0; i < 9; i++)
        {
            buttonArray[i] = new Button(" ");
            buttonArray[i].setBackground(Color.gray);
            buttonArray[i].addActionListener(this);
            add(buttonArray[i]);
        }
        addWindowListener(new winClose( ));
        changeColorThread = new Thread(this);
        gameTimeTimerTask = new GameTimeTimerTask(this);
        gameTimeTimer = new Timer( );
        gameTimeTimer.schedule(gameTimeTimerTask, GAME_TIME);
        changeColorThread.start( );
        setSize(300, 200);
        setVisible(true);
    }
    public int getScore( )
    {
        return score;
    }
    public void setStop( )
```

```
{
   stoped = true;
}
public boolean getStop( )
{
   return stoped;
}
public Button getButton(int index)
{
   return buttonArray[index];
}
public void actionPerformed(ActionEvent ae)
{
   if(ae.getSource( ) instanceof Button)
   {
      Button temp = (Button)(ae.getSource( ));
      changeBackToGray(temp, true);
   }
}
synchronized void changeBackToGray(Button changeBtn, boolean clicked)
{
   if(changeBtn.getBackground( ).equals(Color.red) && !stoped)
   {
      if(clicked)
      {
         score++;
      }
      changeBtn.setBackground(Color.gray);
      setTitle("붉은색단추를 누르면 1점을 얻는다. 현재 얻은 점수: " + score);
   }
}
public void run( )
{
   int selected = 0;
   while(!stoped)
   {
      do
```

```
            {
                selected = (int)(Math.random( ) * 9);
            }while(changeGrayToRed(selected));
            try
            {
                Thread.sleep(TIME_INTERVAL);
            }
            catch(InterruptedException ie)
            {
                ie.printStackTrace( );
                return;
            }
            changeBackToGray(buttonArray[selected], false);
        }
    }
    synchronized boolean changeGrayToRed(int index)
    {
        if(buttonArray[index].getBackground( ).equals(Color.red))
            return true;
        else
        {
            buttonArray[index].setBackground(Color.red);
            return false;
        }
    }
}
class GameTimeTimerTask extends TimerTask
{
    GameFrame parent = null;
    GameTimeTimerTask(GameFrame p)
    {
        parent = p;
    }
    public void run( )
    {
        parent.setStop( );
        parent.setTitle("시간이 되였다! 최종점수: " + parent.getScore( ));
```

```
        }
    }
    class winClose extends WindowAdapter
    {
        public void windowClosing(WindowEvent we)
        {
            we.getWindow( ).dispose( );
            System.exit(0);
        }
    }
```

8.2.22. 한 행의 문자를 견본으로 현시하고 견본문자 아래에 같은 모양으로 입력하는 타자련습프로그람을 작성하시오. 문자입력이 틀리면 되돌아와서 다시 입력한다. 한행의 문자를 정확히 입력하면 10점을 얻는다. 지정된 시간에 얻은 점수를 합계하시오.

# 제9장. 망프로그람작성

## 9.1. 련습

9.1.1. 하나의 TextField와 Label을 포함하며 TextField는 사용자가 입력한 주콤퓨터이름을 접수하고 Label은 주콤퓨터의 IP주소를 현시하는 도형대면의 Application프로그람을 작성하시오.

원천프로그람 ch9_e9_1.java

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
public class ch9_e9_1
{
   public static void main(String args[ ])
   {
      new MyFrame( );
   }
}
class MyFrame extends Frame implements ActionListener
{
   TextField hostInputTfd = new TextField(10);
   Label ipOutputLbl = new Label( );
   MyFrame( )
   {
      super("주콤퓨터의 IP");
      ipOutputLbl.setText(" ");
      add(hostInputTfd, BorderLayout.NORTH);
      add(ipOutputLbl, BorderLayout.CENTER);
      pack( );
      show( );
      hostInputTfd.addActionListener(this);
      addWindowListener(new winClose( ));
   }
   public void actionPerformed(ActionEvent ae)
   {
```

```
        if(ae.getSource( ) == hostInputTfd)
        {
          try
          {
            InetAddress ip = InetAddress.getByName(
            hostInputTfd.getText( ).trim( ));
            ipOutputLbl.setText(ip.toString( ));
          }
          catch(UnknownHostException uhe)
          {
            ipOutputLbl.setText("지정된 주콤퓨터를 찾지 못함");
          }
        }
      }
    }
    class winClose extends WindowAdapter
    {
      public void windowClosing(WindowEvent we)
      {
        we.getWindow( ).dispose( );
        System.exit(0);
      }
    }
```

**9.1.2. URL객체는 어떤 작용을 하며 거기에 어떤 자료가 포함되는가? URLConnection클라스와 URL클라스의 다른 점은 무엇인가? 기능상 어떤 추가적인것이 있는가?**

URL객체는 URL주소를 표시하며 거기에 규약이름, 주콤퓨터이름, 파일경로이름, 포구번호 등 4개의 자료가 포함된다. URLConnection클라스는 URL클라스와 같이 지정된 URL의 자료를 얻을수 있을뿐아니라 HTTP규약을 리용하여 지정된 URL에 자료를 전송할수 있다.

**9.1.3. 사용자가 입력한 망페지주소를 접수하고 프로그람에 이미 보관된 주소와 호상 비교하여 만일 같다면 그 망페지를 열람기에 현시하는 Applet프로그람을 작성하시오.**

원천프로그람 ch9_e9_3.java

```
import java.applet.Applet;
import java.awt.*;
```

```
import java.awt.event.*;
import java.net.*;
public class ch9_e9_3 extends Applet implements ActionListener
{
   final String presetURL = "http://www.kcc.co.kp";
   TextField inputURLTfd = new TextField(20);
   public void init( )
   {
      add(inputURLTfd);
      inputURLTfd.addActionListener(this);
   }
   public void actionPerformed(ActionEvent ae)
   {
      try
      {
         if(ae.getSource( ) == inputURLTfd)
         {
            String str = inputURLTfd.getText( ).trim( );
            if(str.equals(presetURL))
            {
               URL myURL = new URL(str);
               getAppletContext( ).showDocument(myURL);
            }
            else
            {
               showStatus("미안하지만 그 주소를 호출할수 없습니다.");
            }
         }
      }
      catch(MalformedURLException murle)
      {
         showStatus("입력한 망자료주소가 틀립니다.");
         inputURLTfd.setText(" ");
      }
   }
}
```

**9.1.4. 지정된 URL주소의 화상과 음성자료를 호출 및 현시하거나 방영하는 Applet프로그람을 작성하시오.**

원천프로그람 ch9_e9_4.java

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
public class ch9_e9_4 extends Applet implements ActionListener
{
   TextField inputURLTfd = new TextField(20);
   Image myImage;
   public void init( )
   {
      add(inputURLTfd);
      inputURLTfd.addActionListener(this);
   }
   public void paint(Graphics g)
   {
      if(myImage != null)
         g.drawImage(myImage, 100, 100, this);
   }
   public void actionPerformed(ActionEvent ae)
   {
      if(ae.getSource( ) == inputURLTfd)
      {
         String str = inputURLTfd.getText( ).trim( );
         System.out.println(str.substring(str.length( ) - 4, str.length( )) + ",");
         if(str.length( ) > 4 && str.substring(str.length( ) - 4, str.length( )).equals(".gif"))
         {
            System.out.println("match");
            myImage = getImage(getDocumentBase( ), str);
            repaint( );
         }
         else if(str.length( ) > 3 && str.substring(str.length( ) - 3, str.length( )).equals(".au"))
         {
            getAudioClip(getDocumentBase( ), str).play( );
         }
      }
   }
}
```

## 9.2. 보충문제

9.2.1. 거리이름, 거리번호, 아빠트번호, 구역, 시, 우편국을 포함하는 주소클라스를 실현하는 프로그람을 작성하시오. 주소객체를 파일에 넣고 빼는 메쏘드를 작성하시오.

9.2.2. 학생번호, 이름, 성별, 나이, 평균점수를 포함하는 학생클라스를 실현하는 프로그람을 작성하시오. 몇명의 학생정보를 접수하는 도형대면을 작성하시오. 만들어진 객체를 하나의 파일에 보관하고 이외에 단추를 설계하여 그 단추를 누를 때 파일에 현재 있는 학생객체를 평균점수가 높은 순서에 따라 배렬한 후 파일에 다시 쓰시오.

9.2.3. 주소객체를 학생클라스중의 하나의 마당으로 되도록 추가한 새로운 학생클라스에 근거하여 2번문제를 수정하고 자기의 코드설계가 좋은 확장가능성을 가지고있는가를 말해보시오.

9.2.4. 아래의 프로그람을 읽고 그 기능을 설명하시오. 그리고 빠진 명령을 보충하시오.

```
import_______________;
import_______________;
import_______________;

public class AnimateApplet extends Japplet
{
   final int WIDTH = 300;
   final int HEIGHT = 200;
   final int NUMBER = 18;
   Image imgArray[ ] = new Image[NUMBER];
   Image currentImg = null;
   int nextIdx = 0;

   public void init( )
   {
      URL myURL = null;
      try
      {
         for(int i = 0; i < ____; i++)
         {
            myURL = new URL("http://www.mywebsite.com/images/img000"
            + __________ + ".gif");
            imgArray[i] = ___________;
         }
```

```
    }
    catch(________________)
    {
        System.out.println("ERROR: Invalid URL." + murle.getMessage( ));
    }
    MyThread myThread = new MyThread(____________);
    myThread.start( );
    setSize(WIDTH, HEIGHT);
  }
  public void paint(Graphics g)
  {
    g.setColor(getBackground( ));
    g.fillRect(0, 0, WIDTH, HEIGHT);
    if(currentImg != null)____(currentImg, 10, 10, this);
  }
  public void nextImage( )
  {
    currentImg = imgArray[nextIdx];
    nextIdx = ________;
    ______________;
  }
}
class MyThread extends Thread
{
  private AnimateApplet parent;
  public MyThread(AnimateApplet p)
  {
    super;
    parent = p;
  }
  public void_____________
  {
    try
    {
      while(true)
      {
        _____________;
```

```
                sleep(1000);
            }
        }
        catch(__________)
        {
            System.out.println(ie.toString( ));
        }
    }
}
```

9.2.5. 망에 몇개의 련속화상과 대응하는 음성파일이 보존되여 있다고 하자. 이 파일을 내리적재하는 프로그람을 작성하시오. 국부적인 Applet에서 동화상을 현시하고 반주음을 방영하시오.

9.2.6. 봉사기와 의뢰기란 무엇인가? 의뢰기, 봉사기의 작업방식을 간단히 서술하고 Socket클라스와 SeverSocket클라스의 다른 점을 비교하시오.

9.2.7. 암호화된 망통신을 실현하는 프로그람을 작성하시오. 문자렬을 발송하기전에 Caesar암호화법을 사용하여 문자렬을 암호화하시오. 수신자가 접수한 후 암호를 해득하시오. Caesar암호화법은 원래 자모를 자모순서에 따라 어떤 지정된 위치로 평행이동하여 얻어진 문자를 그 문자의 암호로 한다. 실례로 "lazy"가 한자리 평행이동하면 "mbaz"로 된다.

9.2.8. 간단한 망상에서의 대화체계를 실현하는 프로그람을 작성하시오. 봉사기에서 현재 직결되여있는 사용자이름을 보관한다. 새로운 의뢰기가 봉사기에 련결될 때 사용자이름은 의뢰기에 현시된다. 사용자가 그 가운데서 하나의 대화객체를 선택할 때 봉사기는 물음정보를 내보낸다. 만일 대방이 동의하면 의뢰기를 통하여 대화정보를 전송한다.